commodore

the microcomputer magazine

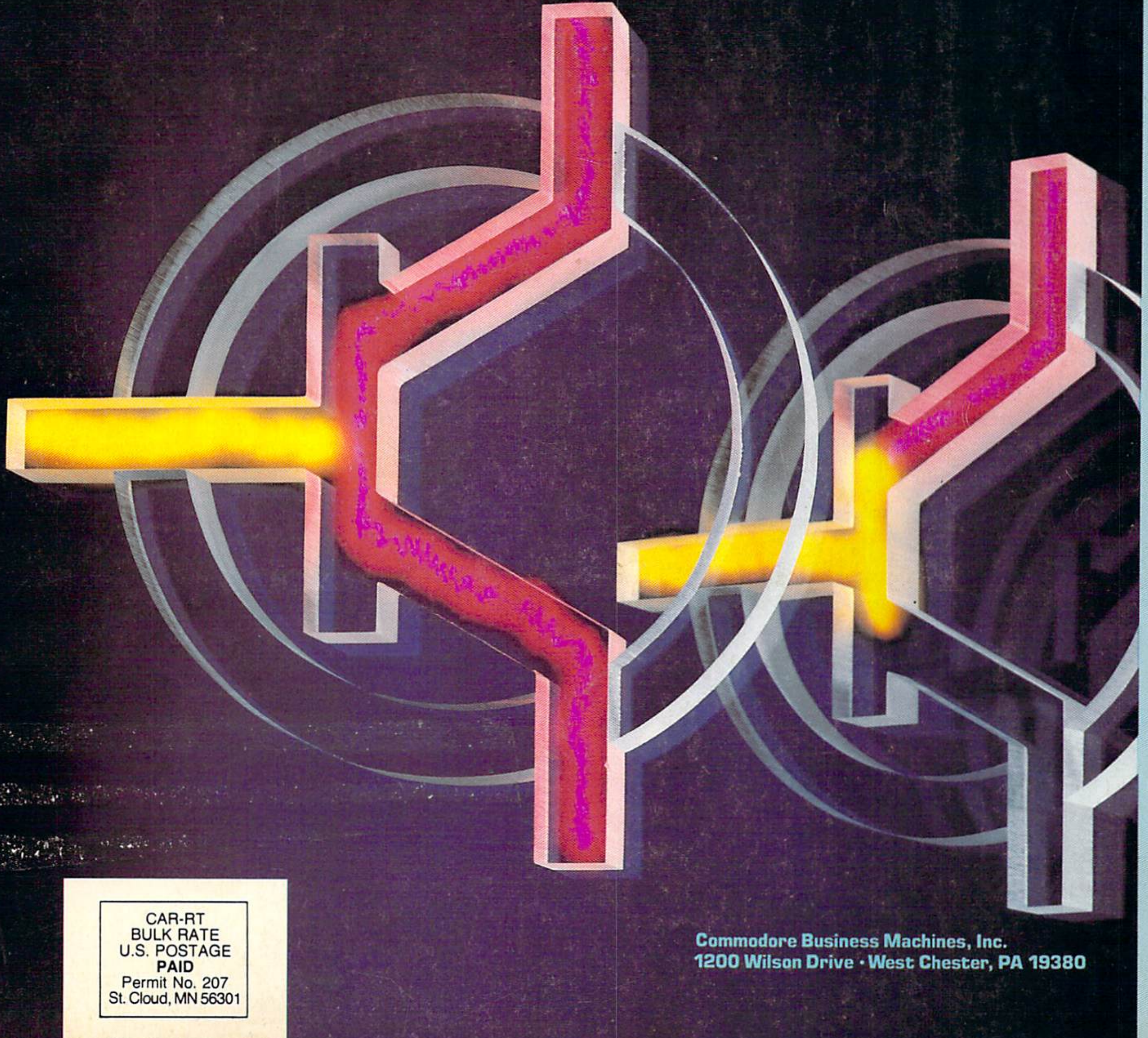
Volume 4, Number 3, Issue 24
$2.50 U.S.
$3.50 Canada

Programming Special

Debugging Can Be Fun!

How Does Your Computer Think?

New Programming Languages for Commodore Computers



CAR-RT
BULK RATE
U.S. POSTAGE
PAID
Permit No. 207
St. Cloud, MN 56301

Commodore Business Machines, Inc.
1200 Wilson Drive · West Chester, PA 19380

Info-Designs announces . . . Small Business Accounting for the Commodore-64

Now the power of Info-Designs Management Accounting System is available on the Commodore-64 in a full and faithful version!

Thousands of these quality business accounting software packages have been sold on the CBM computer at $595 each. Now, similar features are available to the small business user on the new Commodore-64 for $199 per module!

Select the accounting modules you need—

- Accounts Receivable/Billing
- Accounts Payable/Checkwriting
- General Ledger
- Inventory Management
- Payroll
- Electronic Calendar ($149)

Limited Introductory Offer!

Our SoftPack combination contains the "Big-3" accounting—A/R, A/P and G/L—for only $495. Available for immediate delivery!

Flexible Design

The accounting system will work with one or two VIC-1541 disk drives (or 2031/4040 with IEEE interface), 1525 printer, and color or b&w monitor or TV.

Customer Support Plan

As part of Info-Designs ongoing effort to provide the highest quality microcomputer applications in the marketplace, we offer an optional telephone consulting service to support installation and ongoing operations.

Order NOW. . .for immediate delivery

See your local Commodore-64 Dealer or call us directly at **(313) 540-4010**. MasterCard and Visa accepted.

InfoDesigns

6905 Telegraph Road • Birmingham, MI 48010 • (313) 540-4010

You need more than ABC, BPI, MAS, BEC, EBS, XYZ...

Workhorse solutions for tough questions.

There are lots of bookkeeping and business systems for your Commodore computer. Maybe they all seem about the same.

Ours is different: real business software for real computers, with capabilities you need, at a price you can afford.

When professional computer dealers who were dissatisified with their current accounting software were allowed to trade it in for our System III, we were inundated. We got practically everything — BPI, EBS, MAS — just about everyone's.

We have general ledger, accounts receivable, accounts payable, payroll, inventory, mail list management and much more. There are special packages for oil accounting, church records, encumbrance accounting, pharmacy management, etc.

You will like our user-defined reports. You decide what your Balance Sheet, P&L, Budget Analysis, etc., will look like.

If you have purchased a Commodore 64*, you will be pleased when you see our complete line of business software for the '64. This software is no rinky-dink, rip-off software that is an upgrade from the VIC. These are real workhorse programs that use file structures developed for the big boys.

You have purchased the right computer when you bought Commodore. Now do it again. Buy the right software. Buy it from the right place: your professional computer dealer. That's where you'll get the help you need as you start.

Call or write for the name of the dealer nearest you.

Dealer Hotline: 1-800-527-4548

OFFICIALLY APPROVED BY COMMODORE

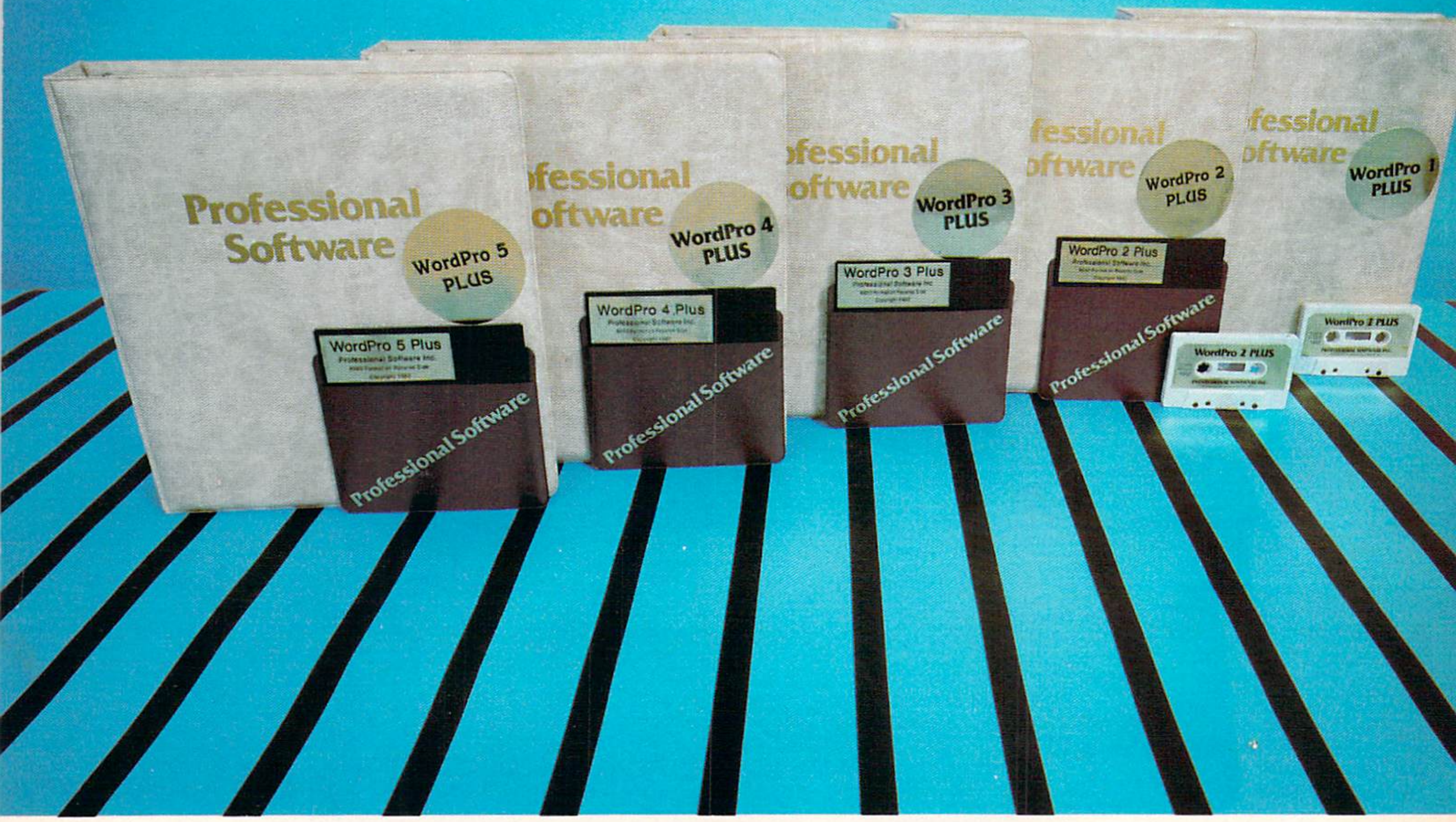
*Commodore 64 is a registered trademark of Commodore

Available in Canada through
Canadian Micro Distributors Ltd.
500 Steeles Ave., Milton, ONT L9T 3P7
Telephone: 416-878-7244 • Telex: 06-961242

Southern Solutions

P.O. Box P, McKinney, Texas 75069 - (214) 542-0278

FOR YOUR COMMODORE WORDPROCESSING NEEDS INVEST IN THE BEST



WORDPRO PLUS. IN A CLASS BY ITSELF.

When choosing a Word Processor for your Commodore™ computer, there's no reason to settle for anything but the best — in a word. . . WordPro™

With over 30,000 happy clients churning out letters and documents all over the world, the WordPro Plus™ Series is unquestionably the #1 selling software package on Commodore computers! So when you choose WordPro, you know you're investing in a trial-tested program that's a real winner. And WordPro is NOW available for your Commodore 64™ computer — at prices starting as low as $89.95.

Designed for the user who has no computer or word processing experience whatsoever, WordPro Plus brings a new dimension to the term "user-friendly." More than just easy to use, WordPro will turn your Commodore computer into a sophisticated time saving word processing tool — loaded with the same inventory of features found in systems costing much, much more.

Our nationwide team of over 600 Professional Software/Commodore computer dealers will help you choose the WordPro Plus system that is best for your needs. Our full-service dealers have been set up to provide strong customer support. In addition to helping you choose the right system, many Professional Software dealers also offer WordPro Plus training and system installation.

Professional Software offers a complete spectrum of WordPro word processing software for Commodore computers ranging from the Commodore 64 to the more business oriented 8000/9000 series computers. And WordPro 4 Plus and 5 Plus also interact with our database management systems including InfoPro and The Administrator. So whatever your Word Processing needs, there's a WordPro system that's right for you.

Invest in the best. . .
WordPro Plus. In a class by itself.

Call us today for the name of the WordPro Plus dealer nearest you.

Professional Software Inc.

51 Fremont Street
Needham, MA 02194

(617) 444-5224
Telex: 951579

WordPro™ and WordPro Plus™ are trademarks of Professional Software Inc.

The WordPro Plus Series was designed and written by Steve Punter of Pro Micro Software Ltd.

Commodore™ and the Commodore 64™ are trademarks of Commodore Electronics, Inc.

Dealer inquiries invited.

Don't let price get in the way of owning a quality printer.

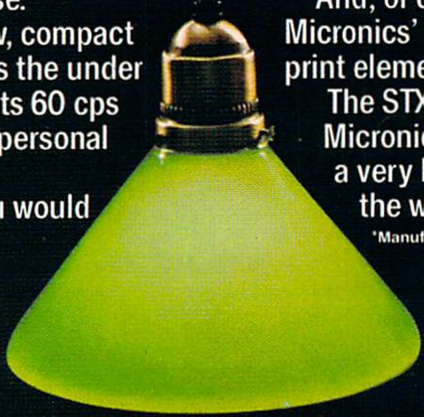
Adding a printer to your computer makes sense. But deciding which printer to add can be tricky. Do you settle for a printer with limited functions and an inexpensive price tag or buy a more versatile printer that costs more than your computer? Neither choice makes sense.

Here's a refreshing option—the new, compact STX-80 printer from Star Micronics. It's the under $200 printer that's whisper-quiet, prints 60 cps and is ready to run with most popular personal computers.

The STX-80 has deluxe features you would expect in higher priced models. It prints a full 80 columns of crisp, attractive characters with true descenders, foreign language characters and special symbols. It offers both finely detailed dot-addressable graphics and block graphics.

And, of course, the STX-80 comes with Star Micronics' 180 day warranty (90 days on the print element).

The STX-80 thermal printer from Star Micronics. It combines high performance with a very low price. So now, there is nothing in the way of owning a quality printer.



*Manufacturer's suggested retail price.

star™
micronics • inc

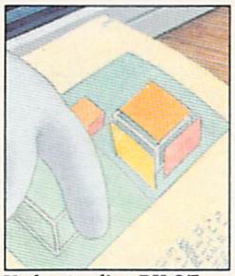
THE POWER BEHIND THE PRINTED WORD.
Computer Peripherals Division, 1120 Empire Central Place, Suite 216, Dallas, TX 75247 (214) 631-8560

The new STX-80 printer for only $199.*

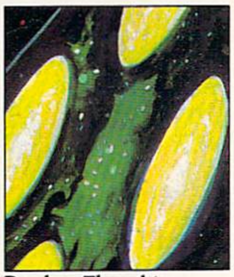
commodore
the microcomputer magazine

features

Volume 4, Number 3, Issue 24


Understanding PILOT

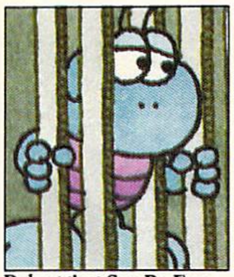
COMAL


Random Thoughts

departments

commodore
the microcomputer magazine

Bits and Pieces


Debugging Can Be Fun

Products for VIC 20® and CBM 64®

SOFTWARE

Word Wizard For The Vic 20® — (Requires at least 8K memory expansion.) A user friendly WORD PROCESSOR with optional joystick control. Easy edit and string manipulation commands that follow the standard format. Full use of function keys for ease of use. 100% machine language with Delete Word, Search functions and Full Justification. Use VIC Graphic printer, or any centronics compatible printer connected to the user port. On tape (support disk). **$34.95.**

ZAP! — Climbing the corporate ladder could be fun except for all that falling paperwork. This Hires arcade type games allows up to 4 players to advance through each floor and change levels to scale the corporate ranks. Be careful, it's easy to be ZAPPED! CARTRIDGE for VIC 20.® **$29.95.**

Bomber Word — A unique graphic word game on cartridge that provides the full thrill of arcade action. Complete with six modes of play options for added enjoyment. Play against the computer or another player. 6 to adult. For VIC 20® . **$29.95.**

Tic Attack — A fast action arcade game on Cartridge that challenges all of your dexterity. Written in machine language for special audio & visual effects. Over 100 levels of play. High score indication. For VIC 20® . **$29.95**

Dot-A-Lot — As you wander through the maze of life collecting Berries, you happen upon some magical fruit. Pick one and the treasures appear, but the Meanies are out today looking to spoil your fun. Defeat them and continue on to a higher level. An ever changing maze plus arcade type animation and sound will provide a real winning CARTRIDGE for the VIC 20® . **$29.95**

Triple Play — Three word games that are both fun and educational. The games that are included are CROSSWORDS (requires at least 8K expansion). Five complete puzzles are included and each puzzle has up to 100 different words. CRYPTO-SOLVE will help you solve those cryptic messages found in newspapers, books, and magazines with a systematic computer technique. Included are approximately 50 different puzzles. You can even enter your own cryptic messages. HIDDEN WORDS will display a matrix of seemingly random letters on the screen. Upon closer inspection, you will be able to find many words. Included are approximately 25 different puzzles. For VIC 20® . **Only $29.95 for all 3**

Keyquest — Our exciting new Arcade type game that takes you through the many levels of an ancient dungeon while gathering treasures and gaining experience points. Monsters, magical keys, and hidden passages all add to the excitement. ON CARTRIDGE for VIC 20® . **$34.95**

Sketch Pad & Char-Gen — A high resolution drawing program that will allow you to save your pictures to tape. Also included is a simple to use character generator that will allow you to design a different character for every printable key. Create game creatures, foreign alphabets, secret symbols or other special characters. One set is included. On tape for the VIC 20® . **$24.95**

HARDWARE

Expand-O-Ram — 16K Expansion Board for the VIC 20® with reset, memory write protect, full memory allocation, plus TWO slots. Like having 2 products in 1. Can be used as a cartridge development system too. **$119.00**

Universal Tape Interface & Duplicator — (Use on the Commodore 64® and VIC 20®). With this device, you can easily load, save or even duplicate tapes with your standard recorder. Full 3 LED indication of Data transfer. A reliable way to Load, Save and Duplicate. NOTE: Duplication requires 2 standard cassette recorders. **Only $49.50**

Tymac Buffered Parallel Cable with Driver — For the VIC 20® & Commodore 64® . This cable assembly plugs into the USER Port and provides a simple and inexpensive way to connect a PARALLEL Printer to your computer. **Only $29.95**

Driver Cartridge for VIC 20® — Take full advantage of the capabilities of your Parallel Printer including full Commodore graphics and formatting. Available for SEIKOSHA, C. ITOH, OKIDATA, and others. Specify printer. **Only $29.95**

Tymac "Connection" — A truly intelligent parallel interface for the VIC 20® and Commodore 64® . It will make your printer operate like the COMMODORE. Printer including graphics, text symbols, tabs, and virtually every other printer function. Plugs into the serial socket. Available for most popular parallel printers. **$119.00**

MasterCard VISA *Dealer and Distributor Inquiries Invited*

NOTE: We solicit hardware and software items for the VIC 20® and CBM 64®. Royalties, license fees, or outright purchases can be negotiated. CBM 64® & VIC 20® are Registered Trademarks of Commodore Business Machines Inc.

micro ware
DISTRIBUTING INC.
1342 B Rt. 23, Butler, NJ 07405
201-838-9027

staff

Publishing Manager
Neil Harris

Editor
Diane LeBold

Technical Editor
Jim Gracely

Staff Writers
Terry Anders
Jeff Hand
Stephen Murri

Contributing Writers
Frank Covitz
Elizabeth Deal
Doris Dickenson
John Heilborn
Walter Herrala
Bruce Jaeger
Charles Knight
Ron Kushnier
Len Lindsay
M.D. Perry
Joe Rotello
Curtis T. Sears, Jr.
David Williams
Mark Zimmerman

Technical Staff
April Koppenhaver
Cyndie Merten
Sarah Meyer
Michael Tomczyk

Circulation Manager
John O'Brien

Circulation Assistant
Kathy Reigel

Advertising Coordinator
Sharon Steinhofer

Graphic Design
Neumann Greenberg Schlenker

Cover Art
Bob Emmott

Printing
Volkmuth Printers

Commodore: The Microcomputer Magazine is published six times a year by the Computer Systems Division, Commodore Business Machines, Inc., 1200 Wilson Drive, West Chester, Pennsylvania 19380. Copyright 1983© by Commodore Electronics Ltd. All rights reserved. No material may be reprinted without permission. Volume 4, Number 3, Issue 24.

Subscription Information: U.S. subscriber rate is $15.00 per year. Canadian subscriber rate is $20.00 per year. Overseas $25.00. To order phone 800-345-8112 (In Pennsylvania 800-662-2444).

Watch for these upcoming issues!

Commodore, Issue 25 (August/September): A special issue on computers in business. Who's doing it and how it's helped them.

Power/Play, Fall: Our adventure game special! An incisive look at mind bogglers like **Zork** and **Deadline**, as well as Scott Adams' adventures and others. Look for us about mid-September.

Advertising Policy

Advertisements within this magazine are presented for the information of our readers. Acceptance of any ad does not constitute an endorsement by Commodore Business Machines. We stand by only those products manufactured by us. In addition, products displaying the "Commodore Approved" logo have been tested by Commodore.

If any product advertised herein fails to perform as advertised, or you are unable to resolve a problem with an advertiser, please bring the matter to our attention. Write to Commodore Publishing Group, 1200 Wilson Drive, West Chester, PA 19380.

TYPRO
DATA MANAGER & WORD PROCESSOR
For COMMODORE 8032 Computer — 8050/4040 Dr.

NOW AVAILABLE FOR THE COMMODORE 64®

DATA MANAGER
Number of records is only limited by your disk capacity. Up to 50 fields per record. Maximum of 75 characters per field. User formatted. Screen editing. Sort and search feature. Pattern match search. Selective field printing and formatting. Form letter addressing. Mailing list and mailing label printing. Format for fanfold Rolodex and index card printing.

WORD PROCESSOR
Screen editing. Automatic line length set. Add, move or delete text. Global edit. Page numbering and titling. Form letter addressing. File append for printing.

BOTH PROGRAMS ABOVE, ONLY $89.00

* Commodore 64 is a trademark of Commodore Electronics, Ltd.

Also for Commodore 64 and 8032
AMORTIZATION SCHEDULE — $30.00 (Disk)
INVENTORY MANAGEMENT — $55.00 (Disk)

All software is fully supported for updates and revisions for up to six months after purchase.
Specify Computer model number and Disk model number.

INPUT SYSTEMS, INC.
25101 S.W. 194 Ave. Homestead, FL 33031 (305) 245-3141
DEALER INQUIRIES INVITED.

GROW WITH US.

VIC 20™ and Commodore 64™ expansion products from Micro Systems Development.

A The Interbus Series. Three interfaces for the VIC 20 and Commodore 64: one for IEEE 488, one for RS 232 and one for Parallel.

The VIE and CIE are IEEE 488 interfaces for the VIC 20. When plugged into the expansion port, the cartridge is "transparent," that is, the user can still attach other peripherals without any interference. Devices such as 4040, 8050, 2031, 2032, 4022 and 8023 can be controlled. The IEEE software can be called by using the 'SYS' command, even in the middle of a BASIC program.

The V232 and C232 are serial interface cartridges which allow connection of various input/output devices such as printers, modems, plotters, etc. to VIC 20 or Commodore 64 computers. Features include: positive and negative voltage swings to meet full EIA standards, straps and jumpers to allow reconfiguration to meet pinouts for any RS232 device, and software selectable reconfiguration such as baud rate, parity, and duplex mode.

The VPI and CPI are parallel interfaces for the VIC 20 and Commodore 64. These interfaces provide direct BASIC use of the parallel printer bus and give "invisible" access to the bus. The VPI can be used only on the VIC 20 and uses the expansion port. The CPI will work with both the VIC 20 and Commodore 64 and does not use the expansion port. The CPI also has switches for setting insertion or deletion of line feed, conversion of Commodore ASCII into standard ASCII or visa versa, addresses printer to device 4, 5, 6 or 7, and allows normally unprintable Commodore characters to be printed in a recognizable form.

B Expandoport Series. Expandoport 3 and Expandoport 6 are three- and six-slot expansion boards for the VIC 20. Each slot on the Expandoport 6 has a switch for controlling power to that connector. The switch allows the use of cartridges which respond to the same memory space. The Expandoport 6 also has a fuse and reset switch. The fuse prevents excessive current drain from the VIC 20 and protects it from 'shorts'. The reset switch allows the user to 'Restart' the VIC 20 without turning power off. This feature allows RAM, which is located in the ROM expansion area, to be protected during 'Restart'.

Expandoport 4 is a four port expansion board for the Commodore 64. It has the same features as the Expandoport 6 and even allows for the use of varying width cartridges.

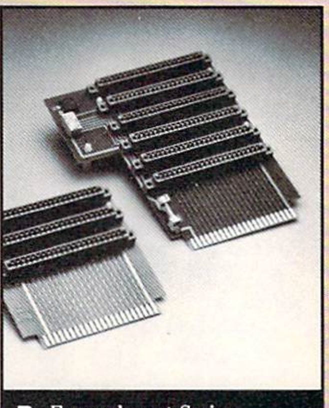
C Terminal Pak Series. The VTE 40 Terminal Emulator (VTE 40) is a hardware and software package which converts the VIC 20 into a 40-column communications terminal. The VTE 40 cartridge is complete. Various set-up parameters such as baud rate, parity, duplex, and bits per character can be selected through a 'menu' format. VTE 40 features are: 40 x 25 text display, user definable communication specs, smooth or normal scroll, print information to printer or disk, generation of control codes, selective omission of data, continuous status line.

The CTE/VTE Terminal Emulator (CTE/VTE) is a software program which converts the VIC 20 or Commodore 64 into a terminal. The user can 'software select' the baud rate compatible with the modem used. Full upper and lower case characters are supported.

D Audio Link. An audio cassette adapter interface for the VIC 20. Features include: use of regular cassette recorders, conversion of VIC 20 digital data to audio and visa versa, normal and inverted cassette signal, remote on/off control and control of external devices.

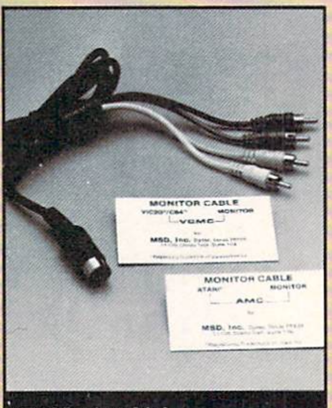
E Monitor Link. This cable assembly allows the VIC 20 and/or the Commodore 64 to interface with a monitor instead of a TV. The Monitor Link provides separate video and audio output. This enables the sound output to go directly into a stereo system for unmatched audio quality. The Monitor Link is great for applications where a TV is not desired. It allows the Commodore 64 user to have high quality resolution on black and white monitors.

A Interbus Series.


B Expandoport Series.

40 x 25 COMMUNICATION TERMINAL CONVERSION CARTRIDGE FOR THE VIC 20 NO EXTRA RAM REQUIRED
VTE - 40
MSD, Inc.
MSD, INC.
C Terminal Pak Series.

D Audio Link.



E Monitor Link.

Call toll free for nearest dealer

1-800-527-5285

Lifetime warranty available upon return of Product warranty card

MSD

MICRO SYSTEMS DEVELOPMENT, INC.

11105 SHADY TRAIL • SUITE 104 • DALLAS, TEXAS 75229

To the Editor:

By now Neil Harris must be "primed out". (See *Commodore,* December/January and May, 1983.) Although the sieve method for finding prime numbers is really fast, it uses array space that ends up as waste. Personally, I prefer to use what I specify in the DIM declaration in the program below. The problem then involves much testing of subsequent values. I get the shakes when I use tests based upon divisibility, as a quick RUN 260 in the program makes very clear.

In a program such as this we like to involve as little testing and calculation as possible for the sake of speed. By generating odd numbers we reduce any divisibility testing by 2. If we generate numbers of the form 6*N+1 and 6*N−1, we eliminate the divisibility testing by 2 and 3. However, as we develop algorithms to eliminate divisibility testing, we expand the program calculations and increase the time consumption. Time consumed decreases on one hand and increases on the other.

```
100 INPUT"NUMBER OF PRIMES ";Z
110 DIMP(Z)::K=5:N=11:P(1)=2
120 P(2)=3:P(3)=5:P(4)=7:P(5)=11
130 READA:IFA=0THENRESTORE:GOTO1
    30
140 J=4:N=N+A:IFK=ZTHEN190
150 J=J+1:Q=N/P(J)
160 IFP(J)>QTHENK=K+1:P(K)=N
    :GOTO130
170 IFQ<>INT(Q)THEN150
180 GOTO130
190 FORI=1TOZ:PRINTP(I),:NEXTI
200 DATA2,4,2,4,6,2,6,4,2,4,6,6,
    2,6
210 DATA4,2,6,4,6,8,4,2,4,2,4,8,
    6,4
220 DATA6,2,4,6,2,6,6,4,2,4,6,2,
    6,4
230 DATA2,4,2,10,2,10,0
240 END
250 REM
260 FORA=1TO10:FORB=1TO10
270 IFB/A*A<>BTHENPRINTA,B
280 NEXTB,A
```

I have chosen a middle ground in which divisibility testing by 2, 3, 5 and 7 are eliminated by DATA increments. The program bashes out 1000 primes in about seven minutes. Moreover, it is nearly as fast as the "sieve" types written by Frens and Ricchezza and it doesn't require 32K to handle 1000 primes. My antique 8Ker is sufficient.

Using 172 primes as criterion, Frens' sieve takes 1170 jiffies, Riccheza's sieve takes 1172 jiffies, and the program below takes 1360 jiffies. The memory requirements of the sieves get horrible as the number of desired primes increases. The program is short and sweet. Is it really the end?

Robert Frens
Kenmore, New York

C

To the Editor:

This note is an appreciation and commentary on Neil Harris' review of the *Word Machine* for the Commodore 64 in the December/January issue of *Commodore.* Brief as it was, the review was essentially accurate and to the point. But there were some data he didn't have that I'll try to supply (I wrote the program, as you probably know).

The one minor error is that Neil implied no correction is possible. The truth is that limited correction is possible, since you can delete back to a block boundary. Therefore, it is usually possible to correct a simple error without editing.

Many of Neil's comments reflect his experience with word processors. If there were a formal definition of "word processor", the *Word Machine* might not even qualify. The *Word Machine* was designed for simplicity. It was not designed to compete with those products in any sense, and was aimed at a completely different market. But it will be interesting to see how the intended users—word processing novices—react to the software. I've used a wide variety of professional products, and even for writing a book I find the *Word Machine* sufficient.

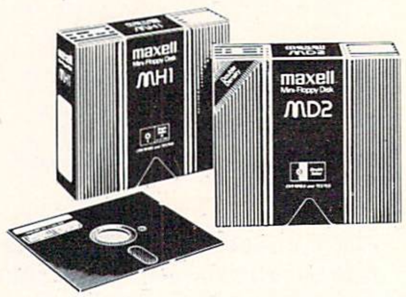
I might also add that the reason we wrote the software in BASIC was to allow the user to personalize the product. The version on which this letter is written has my own letterhead built in. The program is therefore "mine" in at least one sense in which others I have bought are not. Since I use the *Word* and *Name Machines* on a 2001-32 PET with both 2022 and 8300 printers, I have also modified them to allow me to select a printer during initialization. That is typical of the changes a user might want to make that require the program to be in BASIC.

The 1525 printer is a reasonable one for the Commodore 64, but certainly can't produce copy suitable for manuscripts—or anything else demanding a professional word processor. Most of us involved in the design of the *Word* and *Name Machines* recognized the limitations of

the serial hardware and designed the programs to minimize the implications of those restrictions. The people I know who have tried them have said, in effect, that they do their intended jobs better than anything else on the market. It's hard for me to picture a customer who is unsure about which program she needed. The question will answer itself. No professional would use *Word Machine* and no novice will need *Easy Script.* The differences in concept and ease of use appear far more significant than the difference in price.
Michael Richter
Los Angeles, California

maxell

the top of the line



at the lowest price!

Call our Modem Hotline (anytime) - 619-268-4488 for exclusive monthly specials. Our free catalog contains more than 600 fantastic values.

ABC Data Products
(formerly ABM)

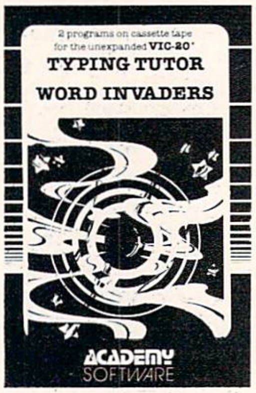
8868 CLAIREMONT MESA BLVD.
SAN DIEGO, CALIFORNIA 92123

ORDERS ONLY	ITT TELEX	INFORMATION
800-850-1555	4992217	619-268-3537

FOR THE VIC 20™

JOIN THE COMPUTER REVOLUTION WITH A MASTERY OF THE KEYBOARD!

In the age of the computer, *everyone* from the school child to the Chairman of the Board should be at home at the computer keyboard. Soon there will be a computer terminal on every desk and in every home. Learn how to use it right ...and have some fun at the same time!



Rated THE BEST educational program for the VIC 20™ by Creative Computing Magazine

TYPING TUTOR PLUS WORD INVADERS — $21.95
(2 programs on one cassette tape for the unexpanded VIC 20™)

Typing Tutor plus Word Invaders makes learning the keyboard easy and fun! Typing Tutor teaches the keyboard in easy steps. Word Invaders makes typing practice an entertaining game. Highly praised by customers:

"Typing Tutor is great!", "Fantastic", "Excellent", High quality", "A source of great joy and learning for our children", "Even my little sister likes it", "Word Invaders is sensational!"

Customer comment says it all . . .
". . . and it was everything you advertised it would be. In three weeks, my 13 year old son, who had never typed before, was typing 35 w.p.m. I had improved my typing speed 15 w.p.m. and my husband was able to keep up with his college typing class by practicing at home."

FOR THE COMMODORE 64™

SPECIAL VERSION OF TYPING TUTOR PLUS WORD INVADERS NOW AVAILABLE FOR THE COMMODORE 64™... $21.95 *(Tape)* $24.95 *(Disk)*

All of the features of the VIC 20™ Version and more

SPECIAL!

SPRITE DESIGNER
by Dr. Lee T. Hill

$16.95*(Tape)* $21.95*(Disk)*

Create and then transform sprites automatically. We have the other sprite making programs, but this is the one *we use* to make sprites. The automatic transformations are great!

VISA Shipping and handling $1.00 per order. California residents add 6% sales tax. VISA and Mastercard orders must include full name as shown on card, card number, and expiration date. Free catalog sent with order and on request. MasterCard

ACADEMY SOFTWARE

P.O. Box 9403, San Rafael, CA 94912 (415) 499-0850

Programmers: Write to our New Program Manager concerning any exceptional VIC 20TM or Commodore 64TM game or other program you have developed.

Reducing Programming Frustrations

We get many phone calls and letters from readers—especially those who are new to computing—who have spent a lot of time typing in programs that they then can't get to run. True, sometimes it's our fault, although that's becoming less and less the case since Jim Gracely joined the staff. (When we do find out we've made a mistake, by the way, we always correct it. Just check out our "That Does Not Compute" department each issue.) However, many times the program won't run because it contains a typo somewhere in its inner recesses that for some reason, no matter how many times you proofread it, you just don't see. And that "syntax error" or "illegal quantity error" message really starts getting on your nerves.

Hopefully, Jim's article on debugging will help you figure out how to locate and correct those kinds of errors —which, as he points out, are actually the easiest kind to find and fix. I can hear some of our more frustrated typists out there snorting their disagreement, but just wait til you have to chase down an analytical error, elusive and annoying as a nasty little "no-see-um" that turns an otherwise delightful beach into a morass of aggravation.

But there's a larger point I'd like to make about programming—whether you're typing in someone else's work or creating your own. And that is that programming, like every other creative activity, is a process that really doesn't have a clear-cut "right" or "wrong" set of answers. Most of us have been trained to think in terms of "right" and "wrong", especially when it comes to mathematics and science, so it's hard for us to grasp the idea that an activity like programming is open-ended—a form of communication, like writing. It just so happens that the communication is between you and the computer. (For some enlightening information about what your computer is "thinking" when you send it communicative signals, see Jeff Hand's article in this issue.)

A program can be approached from many different directions, modified, improved on and played with. The important thing is that you learn from your experience, for better or worse. If, however, you're having too many "worse" experiences, let me make some suggestions. Before you pick up your phone, or take your pen in hand to write someone a nasty letter, see if you can find a book to take you through basic programming procedures. Many bookstores now have a section devoted to computer books, from slim texts for beginners to fat tomes for experienced machine language programmers. And many of those books are written specifically about Commodore computers.

I'd also suggest, as I always do, that you join (or start) a user group. Our rapidly expanding list of Commodore user groups always appears in our magazines, which makes it easy to find a group in your area. Or post a notice on our User Bulletin Board to start one of your own. Some of the larger groups have huge libraries of public domain software, many have newsletters you can subscribe to and most have members with the kind of experience you need when you're in a bind with a program. When it comes to learning about your computer and what it can do, there's nothing that can beat the one-on-one personal contact you get in a user group.

A third alternative for those who want to decrease their programming frustration is to find an independent computer learning center. At the moment we don't have an official list of these centers, although we know many are using Commodore equipment. You may have to do a little investigating to find one in your area, but chances are good that one may be operating near you.

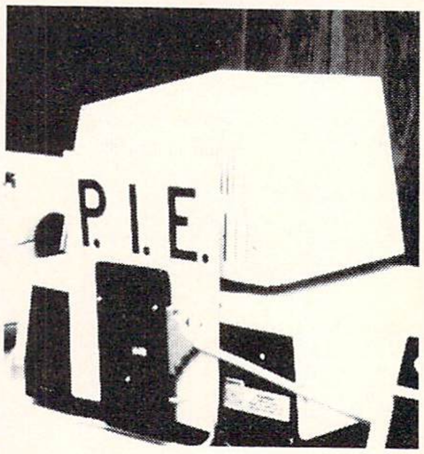
Before I run out of space, just in case you skip "Commodore News" this time, I'd like to point out that Commodore has, finally and forever (?) moved to our new facility in West Chester, Pennsylvania, which is about ten miles down the road from our former quarters in Wayne. So far it's been pretty exciting to have everyone back together under one roof, after we were temporarily scattered over the Wayne-King of Prussia area. Rumor has it that our new building, which also houses our VIC 20/Commodore 64 manufacturing facility downstairs, has about 15 acres under roof. If we're now the "nerve center" for Commodore's North American operations, you can imagine the size of the rest of the nervous system.

Also, please notice that we now have a new Software Division, which is responsible for both creating software in-house and marketing the best from independent developers. You'll be seeing some significant support for your computer coming from these very competent and creative people. Watch for more news on the latest developments in that division. It should be very exciting.

—Diane LeBold
Editor

PIE-C

PET/CBM* IEEE-488 TO PARALLEL PRINTERS By LemData Products



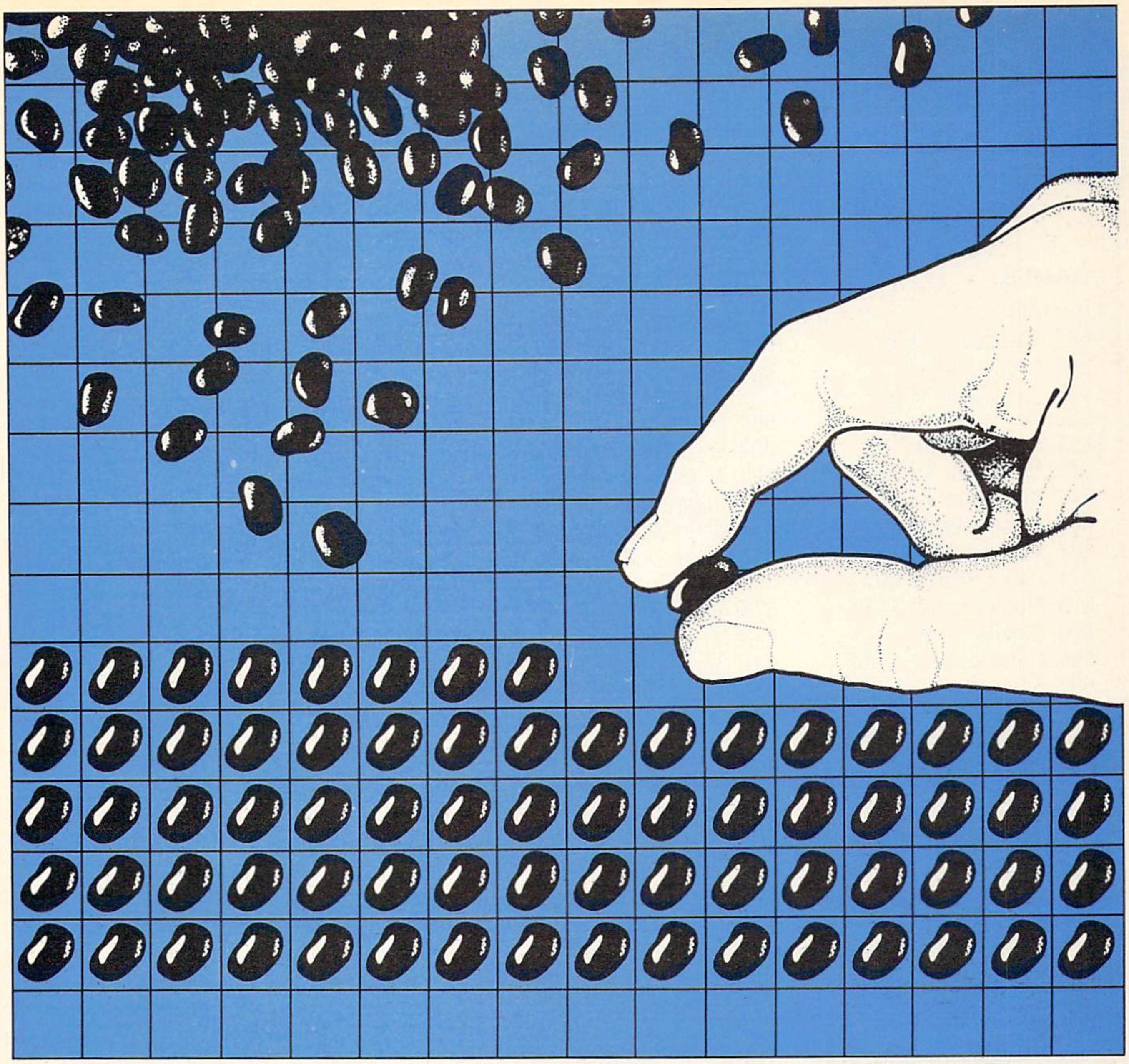
P.I.E.-C MEANS—Professional design, Indispensible features, Excellent quality and Cost effectiveness. You can't buy a better parallel interface for your PET/CBM.

Our P.I.E.-C will interface your PET/CBM through the IEEE-488 bus to . . . . . .

the NEC Spinwriter, the C. Itoh Starwriter, printers by Centronics, Epson, Anadex, Escon Products, the Paper Tigers by IDS, the MIPLOT by Watanabe, the DIP printers, the AJ-841, the OKIDATA printers, plus ALL OTHER parallel ASCII printers.

Assembled with custom case, CBM-TO-ASCII code converter and appropriate cable, the **P.I.E.-C** is only $119.95 (+ $5 S&H). Md. Res. + 5% tax. Specify printer and CBM models.

LemData Products, P.O. Box 1080, Columbia, Md. 21044 Phone (301) 730-3257
*PET/CBM are trademarks of Commodore Business Machines



The BeanCounter™ Sales Management System

The BeanCounter incorporates inventory management, accounts receivable, and order processing functions into a single integrated, on-line, sales management system. It can help you optimize your investment in inventory, reduce your outstanding receivables, and fill customers' orders quickly and accurately.

The system was carefully engineered to ensure continued worry-free operation; in fact, we guarantee it, for one full year. Its state-of-the-art design allows The BeanCounter to offer standard features that aren't even optional on other systems, like bills of material, a mailing label generator, and a word-processor interface.

Information entry and display is simplified through the use of screen "forms" that visually relate to familiar manual systems. You can store more information, and access it faster, with The BeanCounter than any similar system. Specific information can be displayed instantly, or you can choose from a comprehensive selection of informative, timely, reports.

A simple set-up procedure automatically tailors **your** system to **your** business. **You** decide how many inventory items, customers, and invoices are required, and, of course, you can use **your own** part numbers.

The BeanCounter is reasonably priced, and as close as your nearest Commodore dealer. Compare it with any similar system for any micro-computer — we're confident you'll make the obvious choice.

SOFTWERX INC.

SoftWerx Inc.
6174 Quinpool Road
Halifax, Nova Scotia
Canada B3L 1A3

Phone (902) 422-2001

COMMODORE 64K

American Peripherals

GAMES
(on tape)
646 Pacacuda 19.95
650 Logger 19.95
651 Ape Craze 19.95
652 Centropod 19.95
653 Escape 19.95
641 Monopoly 19.95
642 Adventure #1 19.95
648 Galactic Encounter 9.
667 Yahtzee 14.95
671 Robot Blast 14.95
673 Moon Lander 14.95
676 Othello 14.95
686 Horserace-64 14.95
692 Snake 14.95
697 Football 14.95
819 Backgammon 24.95
822 Space Raider 19.95
846 Annihilator 19.95
842 Zwark 19.95
845 Grave Robbers 13.95
841 Pirate Inn Adv. 22.95
904 Shooting Gallery 14.95
816 Dog Fight 19.95
817 Mouse Maze 19.95
818 Ski Run 22.
820 Metro 22.
823 Sub Warfare 29.
838 Retroball 39.95 (cartridge)
839 Gridrunner 39.95 (cartridge)
825 Mine Field 13.
672 Dragster 14.95
662 Oregon Trail 14.95
679 3-D TicTacToe 14.95
655 Castle Advent. 14.95

EDUCATIONAL
(on tape)
644 Type Tutor 19.95
645 Assembly Language Tutor 14.95
687 Fractional Parts 14.95
902 Estimating Fractions 14.95
695 Tutor Math 14.95
870 Square Root Trainer 14.95
699 Counting Shapes 14.95
694 Money Addition 14.95
689 Math Dice 14.95
678 Speed Read 14.95
643 Maps and Capitals 19.95
645 Sprite Editor 19.95
904 Sound Synthesizer Tutor 19.
696 Diagramming Sentences 14.95
690 More/Less 14.95
688 Batting AVERAGES 14.95
802 TicTac Math 16.95
904 Balancing Equations 14.95
905 Missing Letter 14.95
864 Gradebook 15.
810 French 1-4 80.
811 Spanish 1-4 80.
807 English Invaders 16.95
809 Munchword 16.95
812 Puss IN Boot 20.
813 Word Factory 20.
660 Hang-Spell 14.95
905 Division Drill 14.95
906 Multiplic. Drill 14.95
907 Addition Drill 14.95
908 Subtraction Drill 14.95
910 Simon Says 14.95
911 Adding Fractions 14.95
912 Punctuation 14.95

EDUCATIONAL
Series on disk
Computer Science (30 programs) $350
HS Biology (70 programs) $500
HS Chemistry (40 programs) $450
HS Physics (60 programs) $475
HS SAT Drill (60 programs) $99.
Elem. Social Studies (18 pr.) $225
Elem. Science (18 programs) $225
Elem. Library Science (12 pr.) $170
Librarians Package (4 utilities) $110
3rd Grade Reading (20 lessons) $99.
4th Grade Reading (20 lessons) $99.
5th Grade Reading (20 lessons) $99.
6th Grade Reading (20 lessons) $99.
Spanish Teaching (12 lessons) $95.
PARTS OF SPEECH (9 lessons) $95.

BUSINESS
(all on disk)
WORD PRO 3+ 95.00
DATAMAN-64 data base program. 49.95
PERSONAL FILING SYSTEM (index card style) 19.95
HOME FINANCE 19.95
CYBER FARMER $195.
GA 1600 Accounting System 395.
PERSONAL TAX 80.
ACCOUNTS RECEIVABLE 22.
New York State Payroll 89.
MAILING LIST 24.
Manufacturing Inventory 59.
Stock Market Package 39.
Finance 16.95

Visa — Mastercharge — C.O.D.

ORDERING BLANK
To: American Peripherals
122 Bangor Street
Lindenhurst, NY 11757

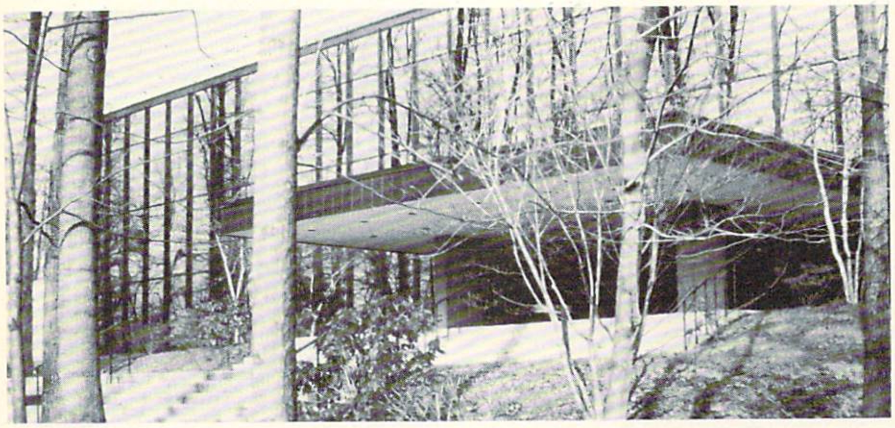
Ship to: Name ______________________
Street ______________________
Town, State, ZIP ______________________

☐ Please send your complete 64K catalog.

ITEM	DESCRIPTION	PRICE

NY State Residents only add 7¼% tax ______
Shipping $1.50
(If COD, add 1.50) ______
TOTAL AMOUNT ______
If Canada or Mexico, add an additional $2.00

Commodore 64™ is a registered TM of Commodore Business Machines Inc.


Commodore's new facility in West Chester, Pennsylvania, features a reflective glass facade that blends into the sylvan setting. Photo by Mel Grauel.

We've Moved!

Commodore is happy to announce that it has officially moved into its new 560,000 square-foot facility just outside West Chester, Pennsylvania—about ten miles from its former quarters in Wayne. The company had already begun manufacturing VIC 20's, Commodore 64's and selected software at the West Chester location some months ago, while waiting for the office facilities to be remodeled. In April, when most of the remodeling was finished, the rest of the company began a gradual move that took about a month to complete.

The new location is now the "nerve center" for the company's entire North American operations, and presently accommodates over 500 Commodore employees. It houses the principle U.S. manufacturing facility for VIC 20's and Commodore 64's and the major U.S. distribution center for all Commodore products. In addition, all offices for both Commodore Business Machines, Inc. and Commodore International Ltd. are here, as well as the newly formed Commodore Software Division (see story following), formerly located in King of Prussia, and the semi-conductor system design group formerly in Norristown.

Sigmund Hartmann, President of Commodore's new Software Division.

Commodore Forms New Software Division

Commodore recently announced the formation of a Commodore Software Division and named Sigmund Hartmann, formerly of TRW, Inc., president. The new division will develop and market a wide range of software for Commodore computers, from games to small business applications.

Commodore sold more computers in 1982 than any other computer company in the world and continues to lead the field worldwide. Since its million-plus end users—many of them computer novices—need top quality software from a reliable source it seems only logical for Commodore to take on that responsibility to a greater degree than ever before. The new Software Division will help do this by establishing a broad base of small business, educational, home and recreational software to better meet the needs of Commodore's many (and varied) users, and to provide the convenience of "one-stop shopping".

Although many software products will be developed in-house, the company will also continue to work with third-party developers by inviting them to become partners with Commodore. The company has already launched a major software acquisition effort to contract existing quality software and is exploring new marketing techniques.

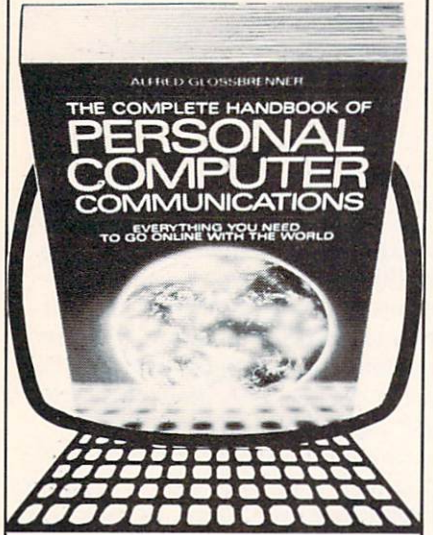
Commodore Begins Shipping "B" Series Advanced Business Computer

The much-anticipated "B" series advanced business micros are now being shipped to Commodore dealers. Available in eight different configurations, these new computers give you state-of-the-art computing capabilities at (as always with Commodore products) affordable prices.

All the computers in the series have the following features in common:

- 80-column by 25-line screen display
- Separate calculator keypad for quick computation
- Ten pre-defined function keys
- A total of twenty easy-to-define

Bring thousands of dollars worth of information home for only $14.95!


ALFRED GLOSSBRENNER
THE COMPLETE HANDBOOK OF
PERSONAL COMPUTER COMMUNICATIONS
EVERYTHING YOU NEED TO GO ONLINE WITH THE WORLD

"The best reason to buy a computer. Packed with useful information—it's an absolute must for computer owners everywhere."
—Gary G. Reibsamen, Vice-President of NewsNet Inc.

Your passport to the electronic universe—how to turn any personal computer or word processor into a communication/information machine of incredible versatility and power, even without a knowledge of programming. Here's just a sampling of what you get—✻ **How to select a modem and communications software** ✻ How to send electronic letters, telexes and MailGrams™ ✻ **Money-saving techniques for using The Source, Dow Jones News/Retrieval, DIALOG, BRS and more** ✻ Get *free* software! Download thousands of programs ✻ **Get privileged information from $30,000 worth of newsletters** ✻ Get discounts of 10% to 40% by shopping on-line ✻ **Trouble-shooter's guide to problem-free communications** ✻ Hundreds of *free* computer bulletin boards (CBBS) ✻ **Access vast libraries (Library of Congress, *N.Y. Times,* magazines, and whole encyclopedias)** ✻ Get real-time computer games and tournaments ✻ **Electronic banking, investment management, and then barter** ✻ Get toll-free numbers, valuable short-cut commands ✻ **Get on-line advice from experienced users of your equipment.**

Plus much more! In just two days you'll see what this unique handbook can do for you—but we'll give you two weeks to decide. Send for your copy *now*; if you're not satisfied return it in two weeks for a full refund.

Main Selection of The Small Computer Book Club Alternate Book-of-the-Month Club/Science Selection

Please return this coupon to your bookseller or direct to:
St. Martin's Press
175 Fifth Avenue, New York, N.Y. 10010 Attn: ML CD

Please send me_____copy(ies) of *The Complete Handbook of Personal Computer Communications* @ $14.95 paperback each. Please add $1.50 per book for postage and handling. My check or money order is enclosed in the amount of $_____. I may examine the book for two weeks and, if not 100% satisfied, I can return it for a complete refund.

Name____________________

Address____________________

City____________________

State________________ Zip________

function keys
- BASIC 4.0
- Expandable memory
- IEEE-488 bus
- RS-232C interface
- Eight-bit user ports
- Integral display
- 6509 microprocessor
- Direct audio output

The features that may vary among the computers in the series are the amount of memory, body style and number of microprocessors. You may choose a computer with either 128K or 256K memory; a high- or low-profile body (with or without monitor, respectively); and with or without an 8088 and/or Z80 microprocessor. The table below shows the possible combinations:

The Commodore "B" series computer, high profile model.

Model	Memory	Standard Microprocessor	Optional Microprocessor	Monitor
CBMX-256-80	256	6509, 8088	Z80	Yes
CBMX-128-80	128	6509, 8088	Z80	Yes
CBM-256-80	256	6509	Z80, 8088	Yes
CBM-128-80	128	6509	Z80, 8088	Yes
BX-256-80	256	6509, 8088	Z80	No
BX-128-80	128	6509, 8088	Z80	No
B-256-80	256	6509	Z80, 8088	No
B-128-80	128	6509	Z80, 8088	No

The addition of Commodore's Z80 microprocessor provides access to CP/M* software. Additionally, the 8088 microprocessor, either built-in or added, provides access to both MS DOS and CC-CP/M-86* software. This extended software capability opens up the many excellent programs written for CP/M and CC-CP/M-86.

In addition, you can customize your system by adding the easy-to-install peripherals available for the "B" series computers. These include Commodore's floppy disk drives and hard disk drives, a variety of printers, modems for telecomputing and monitors for the low-profile machines. **C**

The Harmonizer

by M.D. Perry

Use this program with your Commodore 64 to input multi-voice songs. Save them and play them again and again. You can input your own songs, or send $4.00 to the author for a disk that contains four tunes already entered. His address is 264 Soledad Drive, Monterey, CA 93940.

This user-friendly program for the Commodore 64 permits the user to input multi-voice songs using the notes read directly from sheet music. The user can select the wave form, ADSR (attack, decay, sustain, release) and high and low pulse (if applicable) for each voice prior to entering the notes of the song.

Notes are entered one voice at a time for as many 16-beat measures as desired (in this case a "beat" is equivalent to a sixteenth note). When all three voices have been brought up to the same measure, the music can be played back and edited. If the music is to your satisfaction at this point you can use the ADD option to enter more notes, or you can write the partial song to disk.

Notes are entered by inputting the note, octave number and duration number. The following notes can be entered: C, C#, D, D#, E, F, F#, G, G#, A, A#, B. The duration number represents the number of 1/16's in the note. So a half note is 8, a quarter note is 4, an eighth note is 2, etc. A dotted note is 1-1/2 times the normal note. A typical entry might be: C#,4,2 (C-sharp, fourth octave, eighth note).

I've included the beginning of a typical sheet music song at the end of this article. It indicates, for the key of C, the notes on the treble and bass staff and the appropriate entries for the computer. Note that the first measure has been brought up to a 16-beat count by using the letter "S" (silence) for the appropriate number of beats. In musical terms you'd call this a rest. When you use the "S", you can use any octave number and the duration number is the beat count of the rest.

To use The Harmonizer, follow this procedure.

Step 1: Load and run the program.

Step 2: Press "I" to go to the initial song input mode.

Step 3: Decide on the wave form and ADSR. For starters I'd suggest using 17,63,240 for voice 0; 33,16,16 for voice 1; and 33,16,16 for voice 2.

Step 4: You are now ready to enter notes for voice 0, the treble staff. Look at the sheet music and enter the two measures under V-0. There will be a pause after you enter the first note (S,4,12 RETURN). Note that the computer keeps track of the measure for you. If the number is not a whole number at the end of the measure, you have made an error.

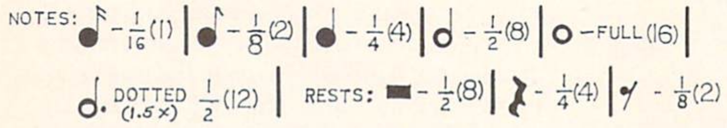
Step 5: When you have entered the two measures under V-0, enter 0,0,0 RETURN to go to the next voice. Enter the two measures under V-1 and again enter 0,0,0 RETURN to go to V-2. Upon completion of the two measures of V-2, enter 0,0,0 RETURN once again and you will be returned to the menu.

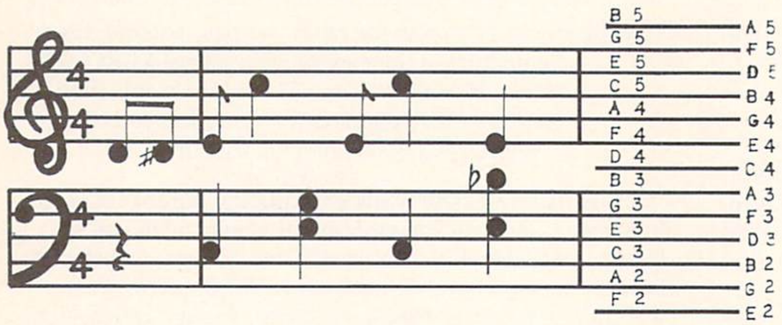
Step 6: Press "P" to play the notes you have entered. Upon completion of the play you are offered a RECORD option. It is not necessary to record at this time since the ADD option on the menu lets you return to where you left off entering the song.

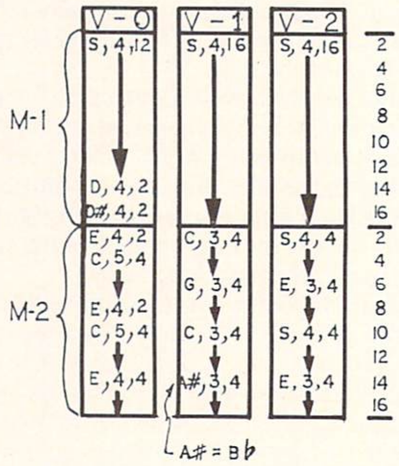
If you make an error while entering the number of beats in voices 1 and 2 so you end up out of sync with voice 0, the computer will recognize the error when you attempt to leave that voice and will inform you of the error. You will then be returned to voice 0 to start the last series over. This can be helpful if you realize you entered a wrong note (pitch and/or octave) after you have already gone on to another. You can then go immediately to the next voice and intentionally make an unbalanced beat entry, which will return you to voice 0 to start the series over so you can make your note correction. Please notice that, because you have to go back to the beginning of the series to make a correction, it's to your advantage to enter only a few measures at a time.

After you have successfully completed a series of entries and have returned to the menu, you may realize you have entered a bad note. At that time you can use the "N" option to enter the note array and total beat number for correction in any voice. Follow the prompts from the computer to overwrite any note. But do not change the duration number or you will be encroaching on the next note. However, it is possible to start at the beginning of a measure and rewrite the measure one note at a time. After each correction you are returned to the menu.

During entry of song notes it is possible to change the wave form of all voices, although it is not possible to change the ADSR at that time. To change the wave forms, complete a note series and go to the menu. Press "C" and enter the new wave forms. But make sure you keep track of the measure number where the change was made, because after you've entered and recorded the song you can use the IF statements in lines 543-44 to change the other parameters (I = 16 times the measure number). Don't forget to delete the REM at the start of each of those two lines. **C**


NOTES: - 1/16 (1) | - 1/8 (2) | - 1/4 (4) | - 1/2 (8) | - FULL (16) |
DOTTED (1.5×) 1/2 (12) | RESTS: - 1/2 (8) | - 1/4 (4) | - 1/8 (2)





```
5 REM: THE HARMONIZER. 'MULTIPLE VOICES 'BY COMMODORE. MODIFIED BY M.D.PERRY
10 POKE53280,6:POKE53281,1:PRINT"[illegible]"
15 S=54272:FORL=STOS+24:POKEL,0:NEXT
20 DIMH%(2,1000),L%(2,1000),C%(2,1000)
30 DIMFQ(11):Z=0
60 FORA=0TO11:READFQ(A):NEXT
70 PRINT"[illegible]MENU"
72 PRINT"[illegible]I[illegible]NPUT NOTES OF SONG"
74 PRINT"[illegible]R[illegible]EAD SONG FROM DISC"
77 PRINT"[illegible]N[illegible]OTE CORRECTION AFTER INPUT"
78 PRINT"[illegible]C[illegible]HANGE WAVE FORM DURING SONG INPUT"
79 PRINT"[illegible]A[illegible]DD NOTES TO SONG":PRINT"[illegible]P[illegible]LAY OR REPLAY SONG"
80 GETA$:IFA$=""THEN80
82 IFA$="I"THEN100
84 IFA$="R"THENGOSUB650:GOTO70
```

```
85 IFA$="C"THENZ=1:GOSUB100:GOTO70
86 IFA$="N"THENZ=3:GOTO750
87 IFA$="A"THENZ=2:I=IM:D=IM:GOTO110
88 IFA$="P"THEN90
89 GOTO70
90 IFIM<>0THEN500
91 GOTO80
100 FORK=0TO2
101 PRINT"▒▒▒▒PARAMETERS FOR VOICE"K"ARE NOW:  W/F-"V(K)",A/D-"D(K)",S/R-"R(K)
102 IFZ=1THENPRINT"▒▒ENTER NEW W/F FOR VOICE #"K":":GOTO104
103 PRINT"▒▒ENTER PARAMETERS FOR VOICE #"K":"
104 INPUT"▒▒W/F(17,33,65)";V(K)
105 IFZ=1THENNEXT:Z=0:RETURN
106 IFV(K)=65THENINPUT"▒▒HI PULSE(0-15)";HP(K):INPUT"▒▒LO PULSE(0-255)";LP(K)
107 INPUT"▒▒A/D(0TO255)";D(K)
108 INPUT"▒▒S/R(0TO255)";R(K)
109 NEXT
110 FORK=0TO2:IFZ=2THENI=D:GOTO112
111 I=0:D=0
112 PRINT"▒▒▒▒VOICE #"K"    ENTER NOTES"
115 PRINT"▒▒ENTER '0' TO GO TO NEXT VOICE"
119 IFZ=3THENZ=0:GOTO70
120 GOSUB800
121 IFK=2ANDI=IMTHEND=IM
122 IFNM=0ANDK<>0ANDI<>IMTHENPRINT"▒▒▒▒ERROR IN ENTRY. REDO LAST SERIES"
123 IFNM=0ANDK<>0ANDI<>IMTHENI=D:IM=D:FORT=1TO2000:NEXT:GOTO110
124 IFNM=0THEN250
135 B=I/16
140 WA=V(K):IFNM<0THENNM=-NM:WA=1
150 DR%=NM/128:OC%=(NM-128*DR%)/16
160 NT=NM-128*DR%-16*OC%
170 FR=FQ(NT)
180 IFOC%=7THEN200
190 FORJ=6TOOC%STEP-1:FR=FR/2:NEXT
200 HF%=FR/256:LF%=FR-HF%*256
210 IFDR%=1THENH%(K,I)=HF%:L%(K,I)=LF%:C%(K,I)=WA:I=I+1:C=I/16
211 IFDR%=1THENPRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒MEASURE"B"TO"C:GOTO119
220 FORJ=1TODR%-1:H%(K,I)=HF%:L%(K,I)=LF%:C%(K,I)=WA:I=I+1:NEXT
230 H%(K,I)=HF%:L%(K,I)=LF%:C%(K,I)=WA-1
240 I=I+1:C=I/16:PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒MEASURE"B"TO"C:GOTO119
250 IFK=0THENIM=I
260 NEXT:Z=0:GOTO70
500 POKES+5,D(0):POKES+6,R(0)
510 POKES+12,D(1):POKES+13,R(1):POKES+15,30
```

```
520 POKES+19,D(2):POKES+20,R(2)
521 POKES+2,LP(0):POKES+3,HP(0)
522 POKES+9,LP(1):POKES+10,HP(1)
523 POKES+16,LP(2):POKES+17,HP(2)
530 POKES+24,15
540 FORI=0TOIM
541 REM-USE LINES 543-544 TO CHANGE A/D & S/R DURING SONG TO BE COMPATIBLE WITH
542 REM-THE W/F CHANGE MADE DURING INPUT.'I' WILL BE 16*(MEASURE #)
543 REM:IFI= ? THENPOKES+5, ?:POKES+6, ?:POKES+12, ?:POKES+13, ?
544 REM:IFI= ?THENPOKES+19, ?:POKES+20, ?:REM-YOU CAN CHANGE PULSES TO IF APPR
550 POKES,L%(0,I):POKES+7,L%(1,I):POKES+14,L%(2,I)
560 POKES+1,H%(0,I):POKES+8,H%(1,I):POKES+15,H%(2,I)
570 POKES+4,C%(0,I):POKES+11,C%(1,I):POKES+18,C%(2,I)
580 NEXTI
589 PRINT"▮▮▮▮DO YOU WISH TO WRITE THE SONG TO DISC?  Y/N "
590 GETA$:IFA$=""THEN590
592 IFA$="Y"THENGOSUB700:GOTO70
599 GOTO70
600 DATA34334,36376,38539,40830
610 DATA43258,45830,48556,51443
620 DATA54502,57743,61176,64814
650 REM READ TO DISC
653 INPUT"▮▮▮▮▮NAME OF SONG";NS$
655 X$=","
660 OPEN2,8,2,"@0:"+NS$+",S,R"
665 INPUT#2,IM
670 FORK=0TO2:INPUT#2,V(K),D(K),R(K),LP(K),HP(K):NEXT
675 FORI=1TOIM:FORK=0TO2:INPUT#2,H%(K,I),L%(K,I),C%(K,I):NEXT:NEXT
680 CLOSE2:RETURN
700 REM WRITE TO DISC
702 X$=","
703 INPUT"▮▮▮▮▮NAME OF SONG";NS$
705 OPEN2,8,2,"@0:"+NS$+",S,W"
710 PRINT#2,IM
715 FORK=0TO2:PRINT#2,V(K)X$D(K)X$R(K)X$LP(K)X$HP(K):NEXT
720 FORI=1TOIM:FORK=0TO2:PRINT#2,H%(K,I)X$L%(K,I)X$C%(K,I):NEXT:NEXT
725 CLOSE2:RETURN
750 REM CHANGE NOTE & OCTAVE
755 PRINT"▮▮▮▮▮THIS SUB-PROGRAM PERMITS YOU TO CHANGE A NOTE LETTER AND OCTAVE."
760 PRINT"▮▮▮▮ENTER VOICE(0,1,OR 2)";:INPUTK
762 IFK>2THENPRINT"▮                                                   ▮▮▮":GOTO760
765 PRINT"▮▮▮COUNT MEASURES BEFORE THE ONE YOU WISH  TO CHANGE"
770 INPUT"▮▮▮ENTER THIS NUMBER";E
775 PRINT"▮▮▮COUNT BEATS IN MEASURE BEFORE THE NOTE  TO BE CHANGED"
```

```
780 INPUT"▒▒▒ENTER NUMBER OF BEATS";F:I=16*E+F
785 PRINT"▒▒▒ENTER NEW NOTE,OCT,DUR # - (DO NOT
    CHANGE THE DURATION #)"
790 INPUT"▒▒▒";NP$,NO,ND
795 GOSUB801:GOTO140
800 INPUT"▒▒N,O,D";NP$,NO,ND
801 IFNP$="0"THENNM=0:RETURN
802 IFNP$="S"THENNM=-128*ND:RETURN
805 IFNP$="C"THENNP=0
810 IFNP$="C#"THENNP=1
815 IFNP$="D"THENNP=2
820 IFNP$="D#"THENNP=3
825 IFNP$="E"THENNP=4
830 IFNP$="F"THENNP=5
835 IFNP$="F#"THENNP=6
837 IFNP$="G"THENNP=7
840 IFNP$="G#"THENNP=8
845 IFNP$="A"THENNP=9
850 IFNP$="A#"THENNP=10
855 IFNP$="B"THENNP=11
860 NM=128*ND+16*NO+NP
865 RETURN
```

A Giant Step
for the computerist

THE PROMQUEEN

Opens up the world of modern electronics. Now – a complete microdevelopment system in a cartridge using the Commodore VIC-20. You get HEXKIT 1.Ø for general purpose 8 bit microprocessor software development, a 4K ROM emulator for testing program in circuits under development plus an EPROM programmer for making hard copy of programs.

All-in-one cartridge with 100 page tutorial manual. **$199.00**

Arbutus Total Soft, Inc., 4202 Meridian, Suite 214, Bellingham, WA 98226. Phone 800-426-1253, in Washington 206-733-0404
Distributed in Canada by **IBC/Distribution Canada,** 4047 Cambie St. Vancouver, BC V5Z 2X9. Phone 604-879-7812

Your Commodore 64™ Deserves An *Assistant*

• The Personal Finance Assistant	$ 59.95
• The Spreadsheet Assistant	$125.00
• The Writer's Assistant	$125.00
• The Filing Assistant	$125.00

RAINBOW COMPUTER CORPORATION

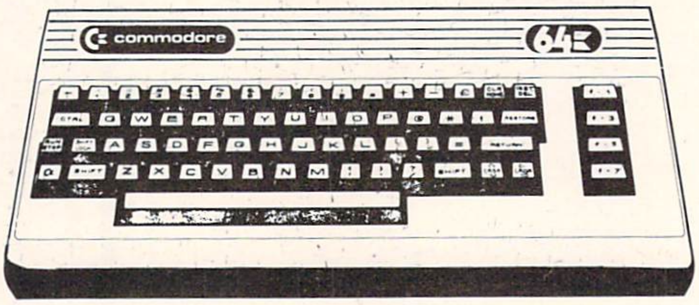
490 Lancaster Avenue
Frazer, PA 19355 (215) 296-3474
Dealer Inquiries Invited

BUSICALC

BUSICALC A Honey of an Electronic Spreadsheet

Why electronic spreadsheet programs?

Electronic spreadsheet programs allow the user to create a gridsheet, spreadsheet, worksheet, or any other table of information, using the memory of the computer as pencil and paper. The computer display or terminal acts as a window through which the user views the information as it is entered. Textual information (such as headings), numerical values, and formulas can easily be entered into the spreadsheet.



For Commodore 64™
For Commodore VIC 20™
For Commodore PET®/CBM™ 40 columns
For Commodore CBM 80 column/SuperPet™

BUSICALC Your Computer Drone for Repetitive Calculations

The outstanding advantage of using a computer is that it acts not only as a pencil and paper but as a perfect eraser and an automatic calculator. The user can quickly and easily make any number of alterations to the data within the table. The BUSICALC will evaluate any formula using the data that has been entered. Further, it retains the formulas and displays the resulting value. With BUSICALC controlling the entry of data, providing a comprehensive memory, and performing arithmetic, the preparation of a spreadsheet is faster and more accurate than if it were prepared by hand.

BUSICALC With the Sting Removed from the Prices

BUSICALC 20	only **$49.00** for the VIC 20
BUSICALC 64	only **$69.00** for the CBM 64
BUSICALC 40	only **$79.00** for the original 40 column PET/CBM
BUSICALC 80	only **$89.00** for the original 80 column CBMs and SuperPets

BUSICALC AVAILABLE NOW FROM YOUR LOCAL DEALER
(800) 227-9998
FOR THE NAME OF YOUR NEAREST DEALER

California, Canada, Alaska and Hawaii please call (415) 965-1735

Skyles Electric Works
231G South Whisman Road
Mountain View, CA 94041

Advanced Bit-Mapped Graphics on the Commodore 64

Part 1

by Frank Covitz

Create bit-mapped graphics in record time with Frank's machine code approach to programming the Video Interface Controller on the Commodore 64.

In this article, we are going to explore in detail high resolution graphics on the Commodore 64. The amazing VIC II (Video Interface Controller), in addition to its various character and sprite modes, has a mode called bit-mapped, which gives us the complete flexibility to control the state of each pixel in a 320 by 200 array. In principle, this is a very easy mode to understand, since for each zero bit in the memory, the corresponding pixel on the screen takes on the background color, and for each one bit, the pixel takes on the foreground color. But to gain this flexibility, we take on a corresponding level of responsibility; that is, we have to control the state of each of the 64000 bits.

We will take on two principle tasks in graphics: first how to turn on or off any individual pixel, given its X and Y coordinates; and second, how to draw a straight line between any pair of X,Y coordinates.

As a secondary, but also important, goal, we would also like to have these routines be fast. The need for speed will become obvious when we discuss the graphics algorithms, but for now you will have to resign yourself to accepting the idea that we will be developing machine language routines to do the work. We will try to make this as painless as possible by giving the corresponding algorithms in BASIC as well.

Finally, to relieve the pain we will link the machine language routines to BASIC so they can be used within BASIC's more friendly programming environment. Note that what we will finally put together is the start of a graphic programming language, and not a stand-alone interactive graphics program.

In considering how to accomplish a complex task, it is usually desirable, if possible, to break up the task into smaller, easier to handle segments. So before jumping in head first, let's outline our objectives in specific steps.

1. Gain access to the bit-mapped mode.
2. Clear the screen to the background color.
3. Given an X-Y coordinate pair, turn the corresponding pixel off or on.
4. Given two sets of X-Y coordinates, draw the best straight line between them.
5. Link the graphic routines so they are easy to call from BASIC.
6. Provide a clean return to normal BASIC, even if an error is made.

To be sure we are talking the same language, let's discuss some of the terms we have been throwing around. In its normal character mode, the Commodore 64 displays characters in a 40-column by 25-row format. Since 40 times 25 equals 1000, 1000 memory locations are needed to represent the screen. You probably are already aware that these locations are normally found at addresses 1024-2023 decimal ($0400-$07E7).

In bit-mapped mode, pixels are displayed in a 320 by 200 format, giving 64000 pixels total. That means we need 8000 memory locations, since each memory location can hold one byte, which in turn is made up of eight bits. Since each bit in this memory controls the state of one pixel on the screen, it is convenient to call it bit-mapped.

So, for example, if we knew the address of a location within the range of the bit-mapped memory, we could turn on eight pixels by POKEing a 255 ($FF) into it. Because of the way this memory is scanned by the VIC chip, this set of eight bits appears as a short horizontal line on the screen. If we had POKEd in a 1, 2, 4, 8, 16, 32, 64, or 128 ($01,$02,$04,$08,$10,$20,$40, or $80) we would see an individual pixel.

You've probably noticed that many times when I've given you a decimal number, I've also given the corresponding hexadecimal number in parentheses. The hexadecimal (hex for short) numbers are preceded by a dollar symbol. It will be extremely helpful to you, and easier once you get the hang of it, to follow the hex representations.

"OK, OK, I follow you so far, so where is this bit-mapped memory already?" you are probably saying. To which I say, "Not so fast, not so fast, we first have to understand a bit more about the VIC chip." Actually you don't really need to understand all the details to be able to use the routines in this article, just as you don't have to understand in detail how the BASIC interpreter works to program in BASIC. But read on; hopefully you will learn some facts and techniques that will definitely be useful for other tasks.

So, a bit of digression before we can even get to step one of our outline. Step zero is a general understanding of how the VIC chip works. The VIC chip can "see" 16K bytes of memory at a time in the same sense that the microprocessor can "see" 64K bytes of memory. The chip, in fact, can see four "banks" of 16K each (see Figure 1), and the bank it sees is determined by two bits of a register in one of the Commodore 64's internal I/O chips. The address of this register is 56576 ($DD00), and the least significant two bits (bits 0 and 1) control the VIC chip bank. The other six bits have to do with the serial bus and the RS-232 port, which we shouldn't disturb. The following table gives the details on the VIC chip banks.

(The xxxxxx means "please do not disturb".)

Let's call these four possibilities banks 0,1,2, and 3, respectively, and try to decide which is the best place to put the bit-mapped memory for the VIC chip to use. Banks 2 and 3 normally contain the BASIC interpreter, I/O locations, and the Kernal operating system, so for now we'll stay away from them. Bank 0 is the one used when you turn the Commodore 64 on, and we could, in fact, use this bank for bit-mapped graphics. However, since BASIC programs normally start using memory at location 2049 ($0801) and expand toward higher memory, we really should avoid bank 0 also (although I actually did use bank 0 in my first trials of high resolution graphics).

This leaves us with a grand total of one bank as the easiest to use for high-resolution graphics, and it gives BASIC a lot of room to "breathe". (NOTE: Bank 2 RAM could actually be used underneath BASIC but that would cause a bit of extra complexity that we don't need yet.)

We are now ready for step one. How do we get VIC to look at bank 1? Easy, just do the following in BASIC (DON'T ACTUALLY DO IT YET!!):

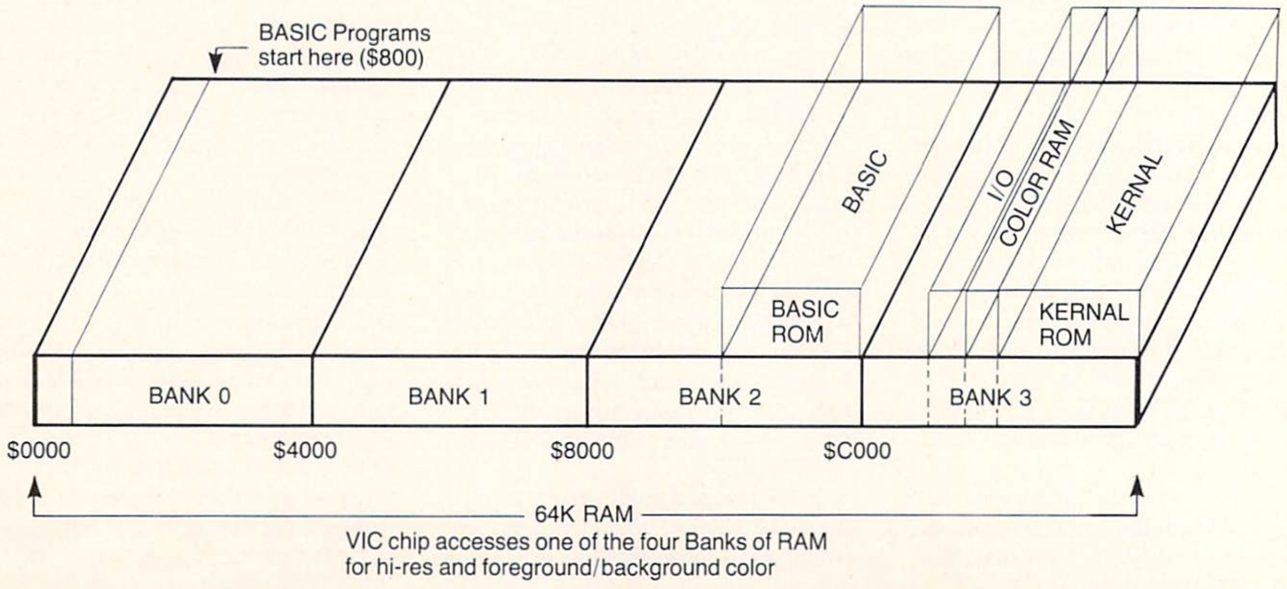
```
10 POKE 56576,(PEEK
(56576) AND 252 OR 2)
```

In machine language, this is equivalent to:

```
LDA $DD00      (get the byte that's
                there)
```

When bits at 56576 ($DD00) are	the VIC chip sees this range of RAM memory	
xxxxxx11	0-16383 ($0000-$3FFF)	Bank 0
xxxxxx10	16384-32767 ($4000-$7FFF)	Bank 1
xxxxxx01	32768-49151 ($8000-$BFFF)	Bank 2
xxxxxx00	49152-65535 ($C000-$FFFF)	Bank 3

Figure 1. Commodore 64 Memory Map


BASIC Programs start here ($800)
BASIC
I/O
COLOR RAM
KERNAL
BASIC ROM
KERNAL ROM
BANK 0
BANK 1
BANK 2
BANK 3
$0000
$4000
$8000
$C000
64K RAM
VIC chip accesses one of the four Banks of RAM for hi-res and foreground/background color

```
AND #$FC        (set bits 0,1 to 0 with-
                out touching the other
                bits)
ORA #$02        (set bit 1 on)
STA $DD00       (VIC nows sees
                $4000-$7FFF)
```

If we had done this, the VIC chip would be scanning bank 1, and we would have a screen full of garbage! (If you actually tried this, you can get back to normal by holding down the STOP key and hitting RESTORE). There's obviously more to do. Actually, two more locations have to be taken care of to get us settled, both located within the VIC chip. Since the VIC chip has 47 internal registers in all, and we are only going to diddle two of them, this really isn't too much to ask. The easiest one to deal with is the control register located at address 53265 ($D011). In it, bit five controls whether we are in bit-mapped mode. A zero here means character mode, and a one means bit-mapped mode. The other bits must be left alone for now. So in BASIC we do:

```
20 POKE 53265,PEEK
(53265) OR 32
```

In machine language this is just:

```
LDA $D011       (get what's there)
ORA #$20        (turn on bit 5)
STA $D011       (we're now in bit-
                mapped mode)
```

The other register we need to set up is the memory pointer register located at 53272 ($D018). When we're in character mode (the normal one) the high order four bits (if eight bits are a byte, then four bits must be a nybble, and two bits should be a nyp, right?) control in which of sixteen 1K blocks the video matrix (the screen) is to be, while the low order nybble determines where the character base is (where the VIC chip picks up the data that form the character).

In bit-mapped mode, these nybbles are used also, but in a different sense. The high order nybble still forms a base address for a 1K piece of memory, but this memory is no longer used to hold character codes; instead it is used to set the foreground and background color for pieces of the bit-mapped memory (more on this later).

Bit 3 (the most significant bit of the lower nybble) is used to determine which half of the 16K memory bank (remember we selected bank 1) that the VIC chip displays as a bit-map. For our memory scheme, we will use the higher half of bank 1 for the 8000 byte high-resolution screen, so we set this bit to a one; the address of the start of the bit-mapped screen is now defined as 24576 ($6000). We want to put the 1K piece of memory close to this (and still in bank 1), so let's go 1K lower to 23552 ($5C00). This is the eighth 1K block in bank 1, so the high nybble of the memory pointer register must contain a seven, right? (Remember, a zero would give us the first block, a one the second block, etc.)

We are now ready... In BASIC:

```
30 POKE 53272, PEEK
(53272) AND 7 OR 120
```

In machine language:

```
LDA $D018       (get what's there)
AND #$07        (wipe bits 7-3)
ORA #$78        (upper nybble =7,
                lower nybble =8)
STA $D018       (simple, wasn't it)
```

To summarize: in step one we set the VIC chip to bank 1, turned on the bit-mapped mode, and fixed the address of both the start of the bit-map and of the corresponding foreground/background memory.

The key addresses so far are:

CBM/PET INTERFACES

The Connecting Links

CmC provides the link to increase your computer's functional ability. The following models come complete with case and cables and have a 30 day money back trial period.

PRINTER ADAPTERS
- addressable-switch selectable upper/lower, lower/upper case
- works with BASIC, WORDPRO, VISICALC and other software
- IEEE card edge connector for connecting disks and other peripherals to the PET
- power from printer unless otherwise noted

RS-232 SERIAL ADAPTER — baud rates to 9600 — power supply included
MODEL-ADA1450a $149.00

CENTRONICS/NEC PARALLEL ADAPTER — Centronics 36 pin ribbon connector
MODEL-ADA1600 . . . .$129.00

CENTRONICS 730/737/739 PARALLEL ADAPTER — 40 pin card edge connector
MODEL-ADA730. . . . .$129.00

EPSON MX-70 PARALLEL ADAPTER — handles graphics — BASIC 4.0 required
MODEL-ADAX70. . . . .$129.00

COMMUNICATIONS ADAPTER — serial & parallel ports — true ASCII conversion — baud rates to 9600 — half or full duplex — X-ON, X-OFF — selectable carriage return delay — 32 character buffer — centronics compatible — power supply included
MODEL-SADI $295.00

ANALOG TO DIGITAL CONVERTER — 16 channels — 0 to 5.12 volt input voltage range — resolution 20 millivolts per count — conversion time less than 100 microseconds per channel
MODEL-PETSET1 . . . .$295.00

US Dollars Quoted
$5.00 Shipping & Handling
MASTERCHARGE/VISA

All prices & specifications subject to change without notice

MENTION THIS MAGAZINE WITH ORDER AND DEDUCT 5% FROM TOTAL

IN THE USA order from:
Connecticut microComputer, Inc.
Instrument Division
36 Del Mar Drive
Brookfield, CT 06804
203-775-4595 TWX: 710 456-0052

IN CANADA order from:
Batteries Included, Ltd.
71 McCaul Street
F6 Toronto, Canada M5T2X1
(416) 596-1405

Dealer Inquiries Invited

bank register	56576 ($DD00)
memory pointer register	53272 ($D018)
control register	53265 ($D011)
start of bit-map	24576 ($6000)
foreground/background	23552 ($5C00)

Before rushing into step two, we need to understand the way the bit-map addresses get displayed on the screen. Turn on your Commodore 64 and put the cursor to the "home" position (or refer to Figure 2). The blinking cursor is now covering the first eight bytes (bytes 0-7) of what will be seen in bit-mapped mode. Byte 0 is the top-most line of the eight short "lines" that form the cursor. The highest order bit of byte 0 is the left-most pixel of that line, which would be set by POKEing a 128 ($80) there. Now space the cursor one position to the right. It now covers bytes 8-15 of what will be seen in the bit-mapped mode. Get the picture? Now, home the cursor and go down one. We are now covering bytes 320-327, right? (since 320 is just eight bytes per character times 40 characters per line). If you understand this you are all set.

What about the foreground/background business? Put the cursor back to home position. Again, this corresponds to the first eight bytes of the bit-map memory. Within this group of bytes the color of the "on" and "off" bits will be controlled by the first byte in the 1K memory that used to be where characters went, but which is now our foreground/background memory (the one we set to start at 23552 ($5C00)). The color of all "on" (foreground) bits within this square are determined by the high nybble of the byte in 23552 ($5C00) and, you guessed it, the lower nybble controls the color of the "off" (background) bits. The colors you get correspond to the ones tabulated in your owner's manual, and are reproduced below:

Figure 2. Memory Map for Hi-Res Screen

'HOME' POSITION	COLUMN 0	COLUMN 1	COLUMN 2	....
	0	8	16	
	1	9	17	
	2	10	18	
	3	11	19	....
Row 0	4	12	20	
	5	13	21	
	6	14	22	
	7	15	23	
	320	328	336	
	321	329	337	
	322	330	338	
Row 1	323	331	339	....
	324	332	340	
	325	333	341	
	326	334	342	
	327	335	343	
	640	648		
	641	:		
	642	:		
Row 2	643	:		....
	644	:		
	645	:		
	646	:		
	647	:		
⋮		:		

Nybble	**Color**
0 ($0)	Black
1 ($1)	White
2 ($2)	Red
3 ($3)	Cyan
4 ($4)	Purple
5 ($5)	Green
6 ($6)	Blue
7 ($7)	Yellow
8 ($8)	Orange
9 ($9)	Brown
10 ($A)	Light red
11 ($B)	Dark grey
12 ($C)	Medium grey
13 ($D)	Light green
14 ($E)	Light blue
15 ($F)	Light grey

In bit-map mode, the ability to control the color within individual eight by eight bit regions is not too useful (at least I haven't found a good use for it), but it has to be taken care of to give us at least a uniform background and foreground color. So, for the start of step one, we do the following in BASIC:

```
40 FOR I=23552 TO
23552+999:POKE
I,80:NEXT I
```

This gives a green foreground color (5*16=80) and a black background (0) to the entire bit-map. Of course, you can pick your own combination, but this is a good one if you are using, like me, a black-and-white monitor (Shocking, isn't it!). In machine code, we do:

```
SETCOL LDA #$50       (set green
                      on black)
       LDX #$00       (initialize X
                      to zero)
SETCL1 STA $5C00,X    (do 1st 3
                      pages)
       STA $5D00,X
       STA $5E00,X
       DEX
       BNE SETCL1     (keep going
                      for 768 bytes)
       LDX #$D8       (do last 232
                      bytes)
SETCL2 STA $5EFF,X
       DEX
       BNE SETCL2
       RTS            (done (leav-
                      ing
                      X = 0))
```

(I hope the little trick at SETCL2 is clear.)

Next, we clear all 8000 bytes of the bit-map memory to the background color (all zeros) in BASIC:

```
50 FOR I=24576 TO
24576+7999:POKE
I,0:NEXT I
```

Note that there are 8000 bytes visible in bit-map mode, not 8K (8192). As far as I am aware, the last 192 bytes are not used or affected in any way by the VIC chip or BASIC, and should therefore be quite safe to use for purposes other than graphics. In machine code, it would be easier to clear all 8192 bytes, but let's be purists and clear only the visible 8000 bytes.

The bit-map clear routine for the 8000 bytes in machine code goes as follows:

```
CLRHR  LDA #$7F       (we're going
                      to clear it
                      backwards)
       STA $FE        ($FD and $FE
                      are safe to use
                      as an indirect
                      pointer)
       LDA #$00       (A will contain
                      a zero from
                      now on)
       STA $FD        (($FD, $FE)
                      now points to
                      first byte of
                      last page of
                      bit-map)
       TAY            (sets Y=0
                      for straight
                      indirect)
       STA ($FD),Y    (this single
                      location needs
                      to be cleared
                      separately)
       LDY #$3F       (this is the
                      offset to reach
                      the last visible
                      byte)
       LDX #$20       (32 pages are
                      involved)
CLRHR1 STA ($FD),Y    (clear a byte)
       DEY            (next one)
       BNE CLRHR1     (go until
                      Y=0)
       DEC $FE        (then go to
                      next page)
       DEX            (X keeps track
                      of total pages)
       BNE CLRHR1     (go until all
                      8000 bytes
                      are cleared)
       RTS            (done, leaving
                      X=Y=0)
```

Since one of the main reasons for going to machine code is to gain speed, the above coding was optimized for speed, and that's the reason for the somewhat less-than-straightforward code. The 8000 bytes are cleared in less than .1 second.

Everything to this point has been really just a preamble to what follows. We are now ready to figure out how to achieve step three. That is, given an XY coordinate pair, where X is in the range 0-319 ($0000-$013F) and Y is in the range 0-199 ($00-$C7), how do we access that individual pixel? What we need is a subroutine which, when given X and Y, returns the address within the bit-map and the data to be POKEd there.

The easiest to figure out is the bit number to be turned on. Since each step in the X direction moves us over one bit and there are eight bits in a byte, the bit number is just the remainder when X is divided by eight. Because eight is an integer power of two, this is simply accomplished by ANDing X with seven. (This isolates the lowest three bits of X.) The result is not quite right since, for example, if X were zero, the bit number would be zero. This would be the rightmost bit in a given bit-map byte, but since we want our X's to go from left to right as X increases, we have to correct this by taking the 'eights complement'. All this means is that we have to subtract the result of our AND procedure from seven. So, we would have in BASIC: BIT=7−(X AND 7). Is this the number to be POKEd into the bit-map to turn on a single pixel? Nope, the correct number is two raised to the BIT power. OK?

Now the big question is which address to POKE into. First, let's agree on where the origin is. The easiest would be to have the XY origin to be at the upper left corner of the bit-map since that is the first memory location (this is in fact the way many home computer graphics do it). However, since everyone knows that the origin should really be at the *lower* left corner, I will use the lower left corner as the origin. All that needs to be done is to subtract Y from 199. So our next operation will be Y1=199−Y.

Next, think about how the bit-map display works—the first eight bytes show up as the upper left square, just where we would expect a character to be if we were in character mode. In fact, every eighth bit-map byte gets us over one character position. So, let's define a variable to tell us which character position we're in. CHAR=INT(X/8) does this for us. Remember, the maximum X can be is 319, and INT(319/8) is 39, right? This means that CHAR goes from 0 to 39, just like the columns of the character mode. As you might expect, the row position is similarly formed by ROW=INT(Y/8). Again, remember that the maximum Y is 199, and INT(199/8) is 24 (character rows go from 0 to 24).

Up to this point, we've been able to calculate which bit is involved and which row and column (same sense as in character mode) we should be in, given an X and Y. Lastly, we must know in which of the eight lines within a character cell we need to be. The answer is LINE=Y AND 7, just like the way we figured out which one of eight bits was required.

We're all set now, since each LINE advances the bit-map address by one, each CHAR advances it by eight, and each ROW (8*40 bytes) advances it by 320. Remembering to add in the address of the first byte of the bit-map, this gives us BYTE=24576+LINE+8*CHAR+320*ROW. Our required BASIC subroutine can now be written as:

```
1000 REM GIVEN X AND Y
CALCULATE ADDRESS AND
DATA FOR BIT-MAP
1010 Y1 = 199-Y
1020 BIT = 2 (7-
(X AND 7))
1030 LINE = Y1 AND 7
1040 ROW = INT(Y1/8)
1050 CHAR = INT(X/8)
1060 BYTE = 24576 +
LINE + 8*CHAR +
320*ROW
1070 RETURN
```

So, to turn on a pixel, given X and Y (first making sure that X is between 0 and 319, and Y is between 0 and 199), we would do GOSUB 1000:POKE BYTE, PEEK(BYTE) OR BIT (since we don't want to disturb any neighboring bits which may have been set). Conversely, you should see that to turn off a pixel at X,Y we would do GOSUB 1000:POKE BYTE, PEEK(BYTE) AND (255-BIT).

We will now tackle the machine code version. I think you probably can already see how to do the additions, logical ANDing and ORing, even dividing by eight (right-shift three times) and multiplying by eight (left-shift three times), since there are direct machine code instructions to do those operations. But what about multiplying by 320 and the two raised to the power? Read on.

The first "trick" we will be using relies on the fact that 8*INT(Y/8) is the same as Y AND 248 ($F8). So we need only to "mask off" the low three bits of Y (or X for that matter) to accomplish this operation. Next, we recognize that 320*INT(Y/8) is the same as (8+32)*8*INT(Y/8), i.e., multiplying a number by 40 is the same as multiplying the number by 32 and then adding eight times that number. Both 32 and eight are powers of two, and multiplying by a power of two is the same as left-shifting the number by that power of two. So we've taken care of most of the 'sticky' operations needed.

Finally, we note that 2 (7−(X AND 7)), which gives us the data to be POKEd could be done by left-shifting a one by (7−(X AND 7)) times. However, to gain a bit of speed, we are instead going to look up the value in a table, which has just eight values in it: 128,64,32,16,8,4,2,1($80,$40,$20,$10, $08,$04,$02,$01). The index into this table is just the result of X AND 7. By doing this, we've eliminated one subtraction and up to seven left-shifts, which is both time- and memory-efficient.

As you may already know, Commodore 64 BASIC, like PET BASIC is very "stingy" on zero-page locations. In fact, the locations $FD and $FE that we used for the indirect pointer in clearing the bit-map are nearly all there are in page zero that BASIC doesn't use. So, since we are going to need more RAM for both the current routine and the next one, let's pick a safe region with sufficient room for our purposes. I've chosen location 828 ($033C) and higher, which is used by BASIC as a cassette buffer (i.e., BASIC uses it only during tape operations) and is therefore safe for holding temporary values while a program is running. So let's define some of the locations we will need:

X.....828 ($033C)	(two bytes(low,high) are needed since X goes up to 320)
Y.....830 ($033E)	(one byte needed for Y)
BIT...831 ($033F)	(this holds the data to be POKEd to the bit-map)
TEMP..832 ($0340)	(two bytes are needed temporarily)

For now, let's assume that the values for X and Y have been set (by POKEing them into 828-830, for example). Our machine code subroutine for setting the byte address (into $FD,$FE) and the BIT data is as follows:

```
PXADDR  LDA #$00   (start by putting 0 into $FE)
        STA $FE    ($FE is the high byte of the bit-map pointer)
        SEC        (get set for subtraction)
        LDA #$C7   (this is decimal 199)
```

```
SBC $033E  (subtract Y,
           leaving
           199-Y in the
           accumulator)
PHA        (and on the
           stack; we will
           need it later)
AND #$F8   (8*INT(Y/8)
           is now in
           accumulator)
ASL A      (multiply by 2)
ROL $FE    (result is now
           in accumulator
           (low) and $FE
           (high))
ASL A      (multiply by 2
           again)
ROL $FE
ASL A      (multiply by
           2 again,
           giving us
           8*(8*(INT(Y/8)))
ROL $FE
TAX        (use x-register
           to hang onto A
           (the low byte)
           for a while)
STA $0340  (also low byte
           held in TEMP)
LDA $FE
STA $0341  (TEMP+1 is
           being used
           to hold the
           high byte of
           8*(8*INT(Y/8)))
TXA        (get low byte of
           8*(8*INT(Y/8)
           back into
           accumulator)
ASL A      (continue mul-
           tiplying by 2)
ROL $FE
ASL A      (once more)
ROL $FE    (we now have
           32*(8*INT(Y/8))
           in A (low) and
           $FE (high))
ADC $0340  (add in low
           byte (note—
           carry is clear
```

DEVELOP-64

FIVE POWERFUL SOFTWARE DEVELOPMENT TOOLS

Plus The Exciting New Book

INSIDE THE COMMODORE 64™

THE BOOK

A complete clear explanation of machine language, Assembly language, Commodore 64 architecture, graphics, joystick and sound effect programming. Detailed step-by-step guide to the use of the development tools. How to combine BASIC and machine language, make auto-start cartridges, interface with the internal ROM-based programs of BASIC and the Kernal. Sample programs fully explained.

THE TOOLS

Assembler/Editor/Loader/Decoder/Monitor
Full-featured Assembler allows use of labels, comments and arithmetic expressions to create machine language programs. Create, save, modify Assembly language programs with the Editor. Load and link machine language modules with the Loader. Decode machine language back into assembly language for study or input to the Editor. Single-step program execution with the Monitor. Combines Assembler/Editor for maximum ease of use.

ALL FOR $54.95 PLUS $2.00 POSTAGE AND HANDLING Add $5.00 for disk version.

Send check, M.O., VISA/MC ($2.00 S.C.) or specify C.O.D. (add $3.00) to:

French Silk smooth ware

P.O. Box 207, Cannon Falls, MN 55009

507-263-4821

Commodore 64™ is a registered TM of Commodore Business Machines Inc.

commodore VIC-20™ Computer

VIC's MOM—MOTHER BOARD EXPANDER

Expand your vic to full limit: provisions for switches in board	3 slot $39.95 / with switches $45.95
VIC-20 to IEEE-488 Interface — allows VIC to use PET/CBM Peripherals.	$79.95
CHARACTER BUILDER—UTILITY AID — will save and load from disk or tape; design your own CUSTOM CHARACTERS; use with any memory configuration	$25.00
ROM EXPANSION BOARD — put your own programs on 2K or 4K EPROM'S; we can put your program in ROM—call for info	$19.50
RS-232 bi-directional — INCLUDES SECOND JOY STICK PORT	$40.00
8K RAM BOARD — can be daisy chained to four	$49.95
NEW SECOND JOY STICK PORT	$20.00
4K RAM BOARD	$39.95
VIC DUST COVER — protect your VIC	$12.95

NEW CARTRIDGE VIDEO GAMES FROM MACHINE LANGUAGE, INC.—SUPER FAST**COLORFUL
AVAILABLE THRU OEM, INC. $25.00

TWO PLAYER GAMES

CLOSE ENCOUNTERS OF THE WORST KIND • BLACK JACK
ACID RAIN • BLOCK BUSTER • DOT GOBBLER • FROGMAN
SPIN TOIDS • CHESS

we need good new machine language games - royalty paid
CALL FOR DETAILS

Completely Tested, 100% Guaranteed

MasterCard VISA

Dealer inquiries invited

TO ORDER CALL 305—464-7549

order from OEM Inc.
2729 South U.S. #1
Ft. Pierce, Florida 33450

Personal checks accepted, allow time to clear—add $2 for shipping—add $5 for COD
Florida residents add 5% state sales tax—Master Card & Visa add 5%

```
             from previous
             step))
STA $FD      ($FD now
             holds the
             low byte of
             40*8*(INT(Y/8)))
LDA $FE      (take care of
             the high part)
ADC $0341    (note—this
             step clears the
             carry flag)
STA $FE      ($FE now
             holds the
             high byte of
             40*8*(INT(Y/8)))
LDA $033C    (get low byte of
             X coordinate)
AND #$F8     (this forms
             8*INT(X/8)
             in the
             accumulator)
ADC $FD      (add to low
             byte of BYTE;
             note that carry
             is already
             clear)
STA $FD
LDA $033D    (take care of
             high byte)
ADC $FE      (this also clears
             the carry flag)
STA $FE      (we now have
             40*(8*INT(Y/8)
             + 8*INT(X/8))
             in $FD,$FE)
PLA          (remember, we
             put Y1 on
             stack, so we
             now fetch it
             back)
AND #$07     (this is LINE)
ADC $FD      (add in to byte
             address; carry
             already clear)
STA $FD
LDA #$60     (finally add in
             bit-map origin)
ADC $FE
STA $FE      (320*CHAR +
             8*ROW +
             LINE + ORI-
             GIN now in
             $FD,$FE)
LDA $033C    (lastly, form
             BIT)
AND #$07     (by ANDing X
             with 7)
TAX          (and using re-
             sult to form
             index)
LDA $0350,X  (fetch from
             table)
STA $033F    BIT now set)
```

This program will allow you to build and animate multi colored sprites on your Commodore 64 without resorting to hours of frustrating use of graph paper and binary numbers. Now you can create in minutes what might have taken hours or even days to see results. Great tool for programmers or fun training aid for novices as well. Contact your local Commodore dealer with this ad or send $24.95 + $2.00 postage - check or money order to:
MAIN MENU Computer Programs
P.O. Box 98005
Seattle, WA 98188

SPRITE BUILDER $24.95

NAME____________________
ADDRESS__________________
CITY_____________________
STATE____________________
ZIP______________________

MAIN MENU
COMPUTER PROGRAMS
P.O. Box 98005
Seattle, WA 98188
(206) 824-3444

For The commodore 64 TM*

*COMMODORE 64 is a registered trademark of Commodore Electronics, Ltd.

```
        RTS          (done)
        ;
        ;here is     BITTAB
        ;
$0350: $80,$40,$20,$10,$08,$04,$02,$01
```

(In case you're interested the whole job took about 150 microseconds!)

Now that everything is set, we can set the pixel on via:

```
SETPIX  LDA ($FD),Y  (get a byte
                     from the bit-
                     map; assumes
                     y-register is 0)
        ORA $033F    (this sets one
                     bit on)
        STA ($FD),Y  (result goes
                     back into
                     the bit-map
                     memory)
```

We can turn a pixel off via:

```
CLRPIX  LDA $033F    (get the BIT
                     value)
        EOR #$FF     ('flips' all the
                     bits)
        AND ($FD),Y  (this turns off
                     one bit)
        STA ($FD),X  (result stored
                     back into
                     bit-map)
```

We also can "flip" the state of a pixel via:

```
FLIPIX  LDA ($FD),Y  (get data from
                     bit-map)
        EOR $033F    ("flip" one bit)
        STA ($FD),Y  (put result
                     back into bit-
                     map memory)
```

I'm sure you've noticed that things have been getting more and more complex as we've proceeded, but if you've gotten this far, you understand the essential features of bit-map graphics on the Commodore 64 (and of bit-mapped graphics in general). Step four, drawing the "best" straight line between two points is next, and believe me, the technique I'm going to use is not the easiest way, but it is one of the fastest, so it will be worth the effort to understand it. We'll pick up there next issue, and conclude our lesson in "advanced" (that is, *fast*) bit-mapped graphics.

BUSINESS & HOME APPLICATIONS FOR VIC-20®

	Quick Brown Fox - the #1 word processor	$60.95
CW-107A	**Home Calculation Program Pack**	48.95
CPV-31	**Data Files** - your storage is unlimited	14.95
CPV-96	**Household Finance Package** - to keep records of all your household expenses	30.95
CPV-208	**Bar-Chart** - display your numerical data	8.95
CH	**Turtle Graphics** - learn programming	34.95
CH	**VIC Forth** - is a powerful language for BASIC programming	49.95
CH	**HES MON** - is a 6502 machine language monitor with a mini-assembler	34.95
CH	**HES Writer** - time-saving word processing tool	34.95
CH	**Encoder** - keep your personal records away from prying eyes	34.95
CT-21	**Statistics Sadistics** - statistical analysis	14.95
CT-121	**Total Time Manager 2.0** - creates personal or business schedules	15.95
CT-124	**Totl Label** - a mailing list and label program	13.95
CT-125	**Totl Text BASIC**	15.95
CT-126	**Research Assistant** - keep track of reference data	17.50
CT-140	**Totl Text Enhanced**	29.95
CM-152	**Grafix Designer** - design graphic characters	12.95
CQ-5	**Minimon** - allows you to program, load, save, or execute machine language programs	13.95
CT-3	**Order Tracker**	15.95
CT-4	**Business Inventory** - to maintain record of inventory	15.95
CS	**Home Inventory** - lists your home belongings	17.95
CS	**Check Minder** - (V-20 & 64) keep your checkbook the right way	14.95
CS	**General Ledger** - a complete general ledger	19.95

BUSINESS & HOME APPLICATIONS FOR C-64

CHC-504	**HES Writer** - word processor	$39.95
CHC-503	**Turtle Graphics II** - utilizes the full graphics of your 64	49.95
CHC-502	**HESMON** - machine language monitor w/mini-assembler	34.95
CHP-102	**6502 Professional Development System**	29.95
CFC	**Data Files** - a management program	27.95
CPV-294	**Calendar My Appointments** - print a calendar for every month in any year	14.95
CPV-296	**The Budgeter** - place your personal finances in order	12.95

MANY MORE PROGRAMS

BUSINESS & HOME APPLICATIONS FOR VIC-20®

CPV-327	**HESCOM** - transfers data and programs bidirectionally between VICs at three times the speed of a disk drive	40.95
CPV-328	**HESCOUNT** - monitors program execution	19.95
CHV	**HESPLOT** - Hi-res graphics subroutines	12.95
CPV-367	**Conversions** - figures, volume, length, weight, area, and velocity to all possible configurations	7.95
CC	**The Mall** - your complete mail program Cassette	24.95
	Disk	29.95

GAMES FOR YOUR VIC-20®

CCS	**Cribbage**	$14.95
CCD	**Motor Mouse**	12.99
CPU-108	**Bomber** - you must decide who you want to fly for, then pick a target and your experience level	9.95
CPU-109	**Amok** - the halls of Amok are populated by robots that obey one instruction - get the intruder	20.95
CPU-153	**Tank vs. UFO** - the tank is moving back and forth along the base, shoot the UFO before it shoots you	9.95
CPU-194	**Snakman** - Pacman for the VIC	14.95
	Defender on Tri - you're the pilot of the experimental ship, Defender	17.95
	3-D Man - the popular arcade game, requires 3K	17.95
	Exterminator - a game full of bugs	20.95

GAMES FOR YOUR COMMODORE 64™

CCS	**Cribbage**	$17.95
CFC	**Flight 64** - what a program! Cassette	14.95
	Disk	16.95
CFC	**Spright Generator** Cassette	15.95
	Disk	17.95
	Mastermind (VIC-20) $12.95 (C-64)	19.95
	Star Trek (VIC)	9.95
	Black Jack (VIC)	11.95
	Tic-Tac-Toe	7.95
	Backgammon	14.95
	Maze 64	15.95

Prices subject to change.

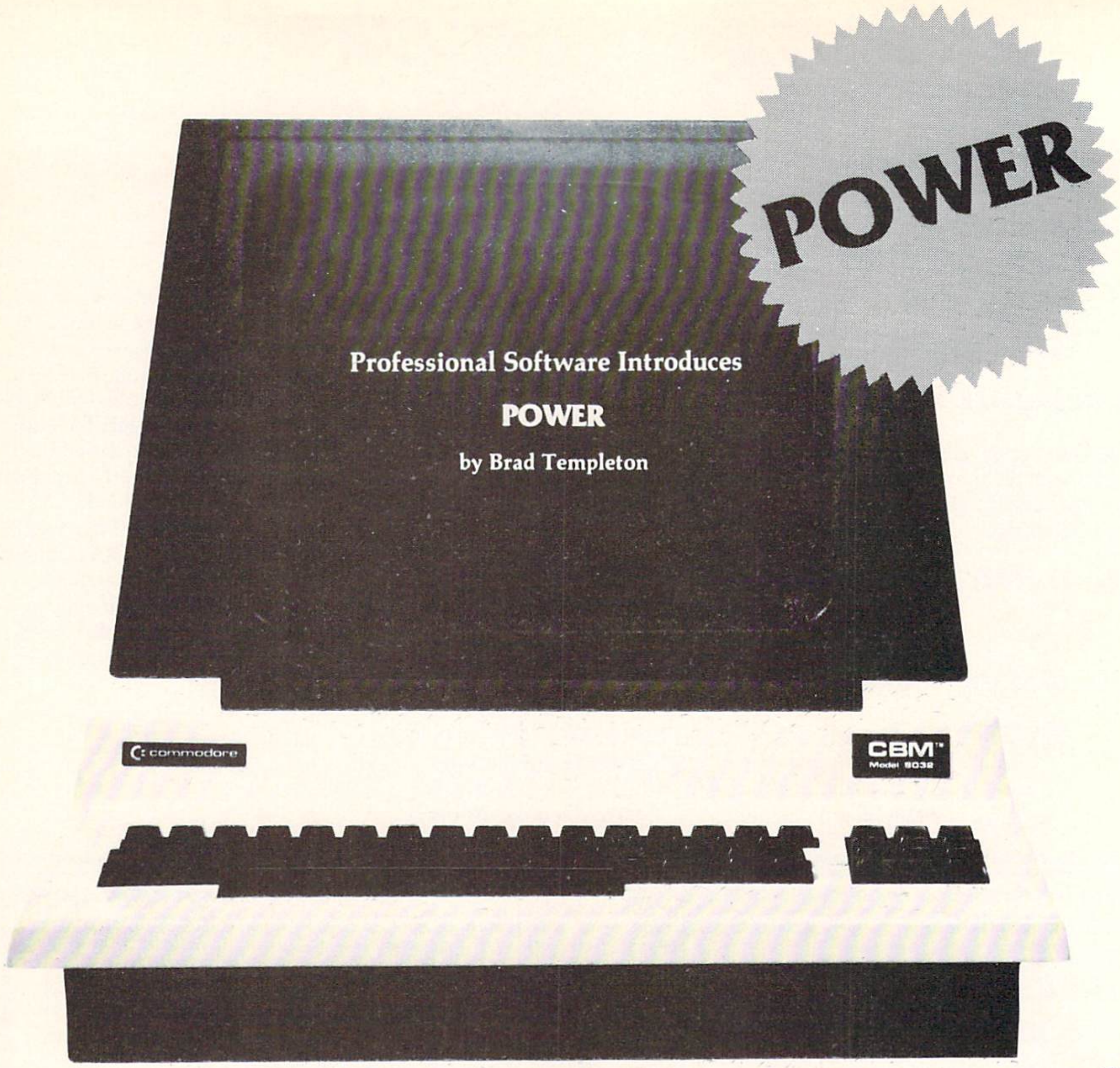
TO ORDER:
P.O. Box 18765
Wichita, KS 67218
(316) 684-4660

MasterCard VISA

Personal checks accepted (Allow 3 weeks) or C.O.D. (Add $2) *Handling charges $2.00*

COMPU ENE

VIC 20 and Commodore 64 are trademarks of Commodore Business Machines, Inc.



ADD POWER TO YOUR COMMODORE COMPUTER

$89.95

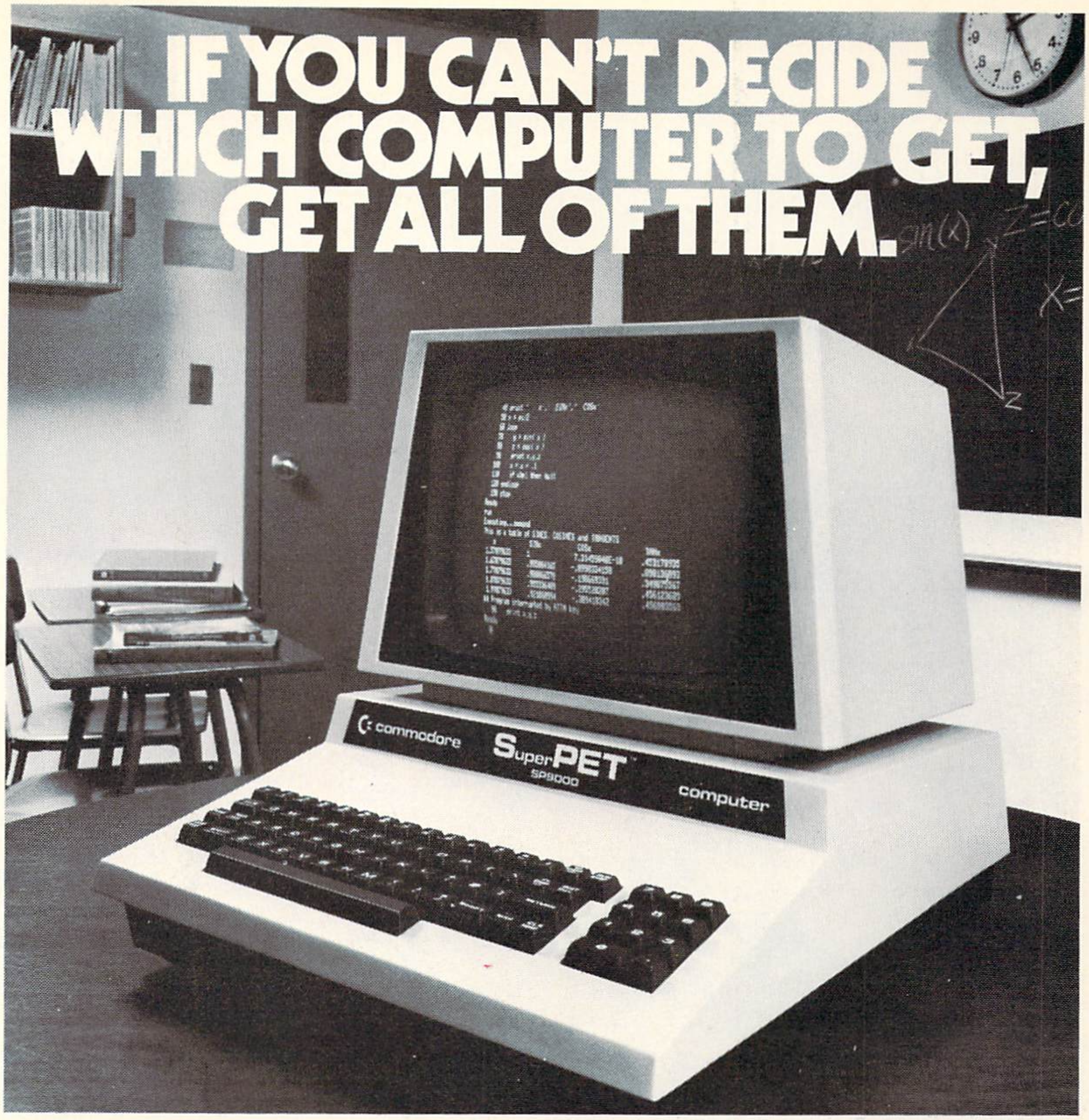
POWER produces a dramatic improvement in the ease of editing BASIC on Commodore's computers. POWER is a programmer's utility package (in a 4K ROM) that contains a series of new commands and utilities which are added to the Screen Editor and the BASIC Interpreter. Designed for the CBM BASIC user, POWER contains special editing, programming, and software debugging tools not found in any other microcomputer BASIC. POWER is easy to use and is sold complete with a full operator's manual written by Jim Butterfield.

POWER's special keyboard 'instant action' features and additional commands make up for, and go beyond the limitations of CBM BASIC. The added features include auto line numbering, tracing, single stepping through programs, line renumbering, and definition of keys as BASIC keywords. POWER even includes new "stick-on" keycap labels. The cursor movement keys are enhanced by the addition of auto-repeat and text searching functions are added to help ease program modification. Cursor UP and cursor DOWN produce **previous** and next lines of source code. COMPLETE BASIC program listings in memory can be displayed on the screen and scrolled in either direction. POWER is a must for every serious CBM™ user.

Call us today, for the name of the Professional Software dealer nearest you.

Professional Software Inc.
51 Fremont Street
Needham, MA 02194
Tel: (617) 444-5224 Telex #951579

CBM is a trademark of Commodore Business Machines, Inc.



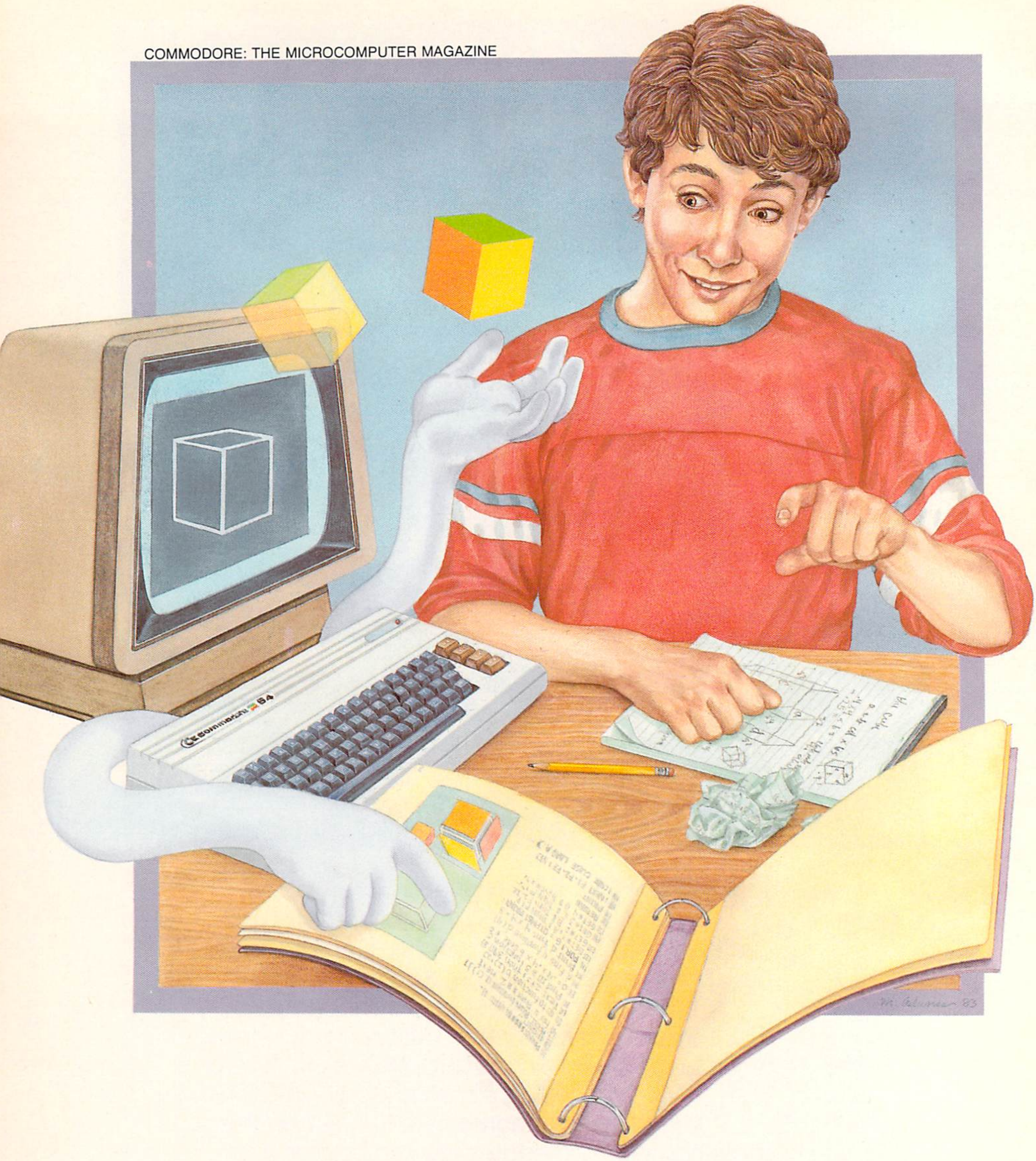
THE SUPERPET.™ 5 LANGUAGES

It's not easy finding a microcomputer that can fulfill all of your requirements. Unless you've discovered the Commodore SuperPET.™

The SuperPET is the *only* microcomputer that comes with five of the most powerful structured languages: MicroBASIC, MicroAPL, MicroFORTRAN, MicroPASCAL, and MicroCOBOL.

We think you'll agree it's far and away the most complete computer you can buy for your school.

commodore
COMPUTER



UNDERSTANDING THE PILOT LANGUAGE

BY STEPHEN MURRI

Teachers can use PILOT for the Commodore 64 to program computer assisted instruction to be used by their students. Here Steve Murri provides a detailed description of how this versatile programming language works.

This article looks at PILOT on the Commodore 64 from a programmer's point of view. First, we discuss PILOT conventions and then cite actual program examples.

The PILOT instruction line consists of several parts. For example:

TS(X>0):This is a sample PILOT instruction

The first part of every instruction is an operation code or "opcode". In this example, the opcode "T" is used for the TYPE instruction. PILOT opcodes are usually only one character long. There are twenty different opcodes in the Commodore 64 version of PILOT. The most commonly used opcodes are T (TYPE), A (ACCEPT), M (MATCH),

ILLUSTRATION—MIKE ADAMS

J (JUMP), and W (WAIT).

Following the opcode, there may be one or more "modifiers" or "conditioners". Modifiers alter the instruction in some manner and are always one character long. In the previous instruction, the "S" modifier alters the TYPE instruction by directing the computer to clear the screen. "Conditioners" determine whether or not the PILOT instruction is to be executed. The conditioner can be any PILOT expression that generates a truth value. In the preceding example, the instruction is executed only if (X>0) is true. Conditioners can also consist of a single digit or letter. The Y and N (yes and no) conditioners determine if the instruction is to be executed based on the result of the last MATCH instruction. In the following example, the Y conditioner is attached to the instruction:

```
TSY:This is a sample PILOT
instruction
```

This instruction is executed only if the last MATCH instruction succeeded. If the N conditioner were used, the instruction would execute only if the last MATCH failed. Single-digit conditioners are discussed later.

The last two parts of a PILOT instruction are the "separator" and the "field". The separator is the colon (:) and must be present in every PILOT instruction. The colon separates the opcode, modifiers and conditioners from the rest of the instruction. The field is to the right of the colon and contains text or PILOT commands.

Interacting With the Computer

We are now ready to look at how PILOT programs are written. The main instructions used for interaction with the user are the TYPE (T:) and ACCEPT (A:) instructions. The following program demonstrates how easily PILOT programs interact with the user:

Note: *Line numbers are not used with PILOT on the Commodore 64. We included line numbers in our examples strictly for explanatory reasons.*

```
 1 D:N$(20)
 2 TS:What shall I call you today?
 3 A:$N$
 4 T:OK, $N$, press RETURN when
   you are
 5 :ready to continue.
 6 A:
 7 T:
 8 :Today our topic is South America.
 9 :
10 W:20
```

Line 1 uses the DIMENSION (D:) instruction to define a string variable (N$) with a maximum length of twenty bytes. Line 2 uses the TYPE instruction with the screen modifier (TS:) to clear the screen and prompt the user for a name. Line 3 uses the ACCEPT (A:) instruction to input the user's name and save it in the string variable N$. Lines 8 and 9 use the continuation (:) instruction. When several TYPE instructions in a row are needed, we do not have to repeat the information in front of the colon; we simply repeat the colon. Finally, the WAIT (W:) instruction in line 10 is used to pause the display for two seconds.

The Power of the Match

One of the most powerful features of PILOT is its ability to interpret exactly what the user is trying to say, regardless of placement or spelling. This is accomplished with the MATCH (M:) instruction. The MATCH instruction performs a "window" search of the user answer buffer and scans for the words or word segments that you specified in the instruction. Spelling errors can be detected by using the "single wild-card character" (*) which matches any character in the corresponding position, or the "multiple wild-card character" (&) which matches any number of characters, including no characters. The MATCH instruction is demonstrated in the following interactive quiz on South American countries:

```
 1 PR:L
 2 *FIRST
 3 TS:Which South American coun-
   try is best
 4 :known for its production of oil?
 5 A:
 6 T:
 7 M:v*n*&u&l*
 8 TY:Correct. It is Venezuela.
 9 CY:R=R+1
10 WY:20
11 JY:NEXT
```

On line 1 we see an example of the PROBLEM (PR:) instruction. The PROBLEM instruction can be followed by an "option list" specifying various options. In this example, the lower-case option (L) is used to convert all user responses to lower case so we may correctly use MATCH.

Line 2 contains an example of a PILOT label. Labels can be one to six characters long and are used as destinations in JUMP and USE (subroutine) instructions. The MATCH instruction, on line 7 uses both the single and multiple wild-card characters. In conjunction with the lower-case option specified in the example, this instruction also matches many possible responses, including any of the following:

Venezuela
The country is venesuela
venizula
VINEZUELO

The instructions on lines 8 through 11 use the Y conditioner and are executed only if the MATCH succeeded. Line 8 TYPES a message acknowledging the correct answer. Line 9 updates the numeric variable "R", which is used to store the number of correct answers. Line 10 pauses for two seconds and line 11 JUMPs to the instruction labelled *NEXT.

True Dialogue

The preceding program example is now expanded to demonstrate PILOT's capability to carry on a true dialogue with the user. To direct the user to the correct answer, the following instructions MATCH for other South American countries:

```
12 M:br*&*1
13 TY:No, Brazil is best known for its
14 :production of rubber and coffee.
15 M:c&il*&
16 TY:No, Chile is known for pro-
   ducing the
17 :finest wines.
18 M:c*l&*b*&
19 TY:No, Colombia is known for
   producing
```

```
20 :the world's finest coffee.
21 M:per!equ!bol!arg!par!ura!chi!braz!col
22 TN:$%B is not a South American country.
23 MY:per!equ!bol!arg!par!ura
24 TY:No, $%B is not the country.
```

Lines 12 through 20 check to see if the user entered Brazil, Chile or Colombia, and if so, informs the user of that country's main product. Line 21 uses the OR (!) operator to MATCH *any* South American country. (In addition to the OR operator, the AND (&) operator can also be specified in a MATCH instruction.) If the MATCH fails, line 22 types out the user answer buffer (%B) and informs the user that this is not a South American country. Finally, lines 23 and 24 determine if the user's response was a South American country other than Brazil, Chile or Colombia; if so, PILOT simply displays the response that this is not the correct country.

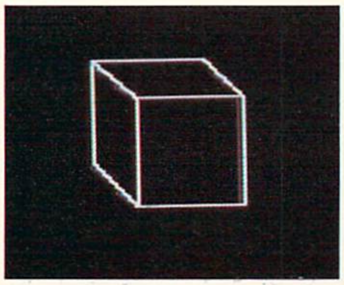
Hinting

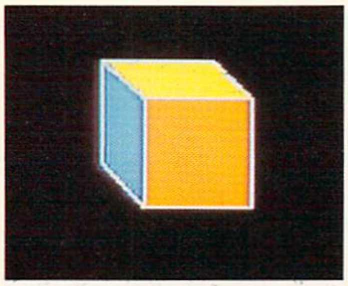
PILOT provides an easy way to present a series of hints through use of the single-digit conditioner. Hints are clues to assist the user in finding the correct answer.

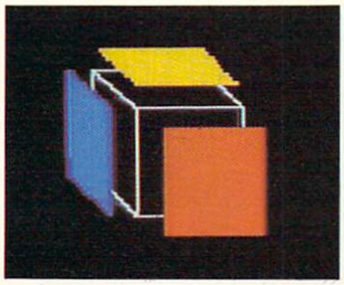
PILOT keeps track of the number of consecutive times the user responds to the same question. This count is stored in what is called the "A" counter. The value of the digit used in the conditioner is compared with the value in the "A" counter. If they are the same, the instruction is executed; otherwise, the instruction is skipped. Our previous program is expanded to include examples of hinting:

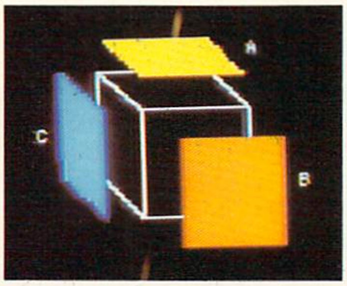
```
25 T:
26 T1:A hint, this country is located at
27 :the top of the continent.
28 T2:Another hint, the capital of this
29 :country is Caracas.
30 T3:The correct answer is Venezuela.
31 W:50
32 J3:NEXT
33 T:
34 :Try again.
35 W:10
36 J:FIRST
37 *NEXT
```

The instructions on lines 26, 28, 30 and 32 use the single-digit conditioner. Line 26 is executed only after the *first* user response. Line 28 types another hint only on the *second* try. Finally, if the user does not have the right answer by the third try, the program types out the correct answer and JUMPs to the instruction labelled *NEXT. Otherwise, a "try again" message is displayed and the JUMP instruction on line 36 sends the program back to the initial question to give the user another chance. The WAIT instruction is used in lines 31 and 35 so the user has time to read the computer responses.


Use PILOT's graphics commands to draw lines.


Next, sprites are combined to form the panels of the cube.


Sprite animation is used to create movement on the screen.



The labels on the panels show PILOT's ability to combine high-resolution graphics and text on the same screen.

PILOT Graphics

One of the most powerful PILOT instructions is the GRAPHICS instruction. The GRAPHICS instruction (G:) is always followed by various graphics commands to specify color, text-cursor positioning, erasing and "windowing". Another set of graphics commands is used to plot and draw on the screen. These commands are always followed by an (X,Y) screen location. When plotting or drawing, the screen is treated like a piece of graph paper with the origin (0,0)located at the bottom left corner of the screen. The commands used to point and draw on the screen are POINT (P), DRAW (D), MOVE (M), FILL (F), UNPOINT (Q) and REMOVE (R).

The following instructions use the GRAPHICS instruction to draw a square on the screen:

```
1 G:Cl;B0;X0;E
2 G:P100,70
3 G:D220,70;D220,150; D100,150;D100,70
```

The first instruction line sets the cursor color to white (Cl), the background and border colors to black (B0;X0), and erases the screen (E). In line 2, the POINT command is used to plot a point at (X,Y) location (100,70). Instruction line 3 creates the square by drawing four lines with the DRAW command.

The square can be expanded into a three-dimensional cube with the following instructions:

```
4 G:D70,90;D70,170;D180,170;D220,150
5 G:M70,170;D100,150
```

Line 4 uses the DRAW command to create the diagonal lines required for depth. Line 5 uses the MOVE (M) command to position the graphics beam at (X,Y) location (70,170) so the final line of the cube can be drawn.

Graphics and text can be combined in the same display as demonstrated by the following instructions:

```
6 G:M225,150;T
7 T:A
```

```
8 G:M227,70;T
9 T:B
```

These instructions label two points on the cube "A" and "B". Line 6 moves the graphics beam to position (225,150). The MOVE command is then followed by the TEXT-CURSOR command (T), which positions the text cursor at approximately the same point as the graphics beam. Line 7 uses the TYPE (T:) instruction to type the letter "A" on the screen. The letter "B" is similarly displayed on the screen with the instructions in lines 8 and 9.

Expanding on our example, the following instructions demonstrate the use of the SPLIT (S) command. The SPLIT command creates windows in any one of five locations on the screen:

```
10 G:S5;B6
11 TS:If the distance between point A
12 :and point B is 12 centimeters
   long,
13 :how many cubic centimeters
   are in
14 :this cube?
15 W:100
```

The GRAPHICS instruction on line 10 (G:S5) splits the screen at the fifth location, creating a window at lines 20 through 25. The background color command (B6) sets the background color of the window to blue. All subsequent TYPE instructions now type text only in this window. If a clear screen is executed, only the text in the newly created window is erased; the original graphics display of the cube is preserved. This window remains in effect until another SPLIT command is executed.

Windows are useful when creating complex graphics displays that are enhanced by several frames of discussion. In the previous example, the window is used to quiz the user on the display. The window area can then be used for the user's response and additional questions.

Programmable Characters

New characters can easily be designed with PILOT on the Commodore 64 by using the NEWCHAR (N:) instruction. The NEWCHAR opcode is followed by the ASCII code number of the character to be redefined. The 8 × 8 bit character definition is then coded as a series of dots which are either "ON" or "OFF". The ON dots are coded with an X and the OFF dots are coded with a period.

The following program creates special characters for "up" and "down" arrows:

```
 1 N:97
 2 ...XX...
 3 ..XXXX..
 4 .X.XX.X.
 5 X..XX..X
 6 ...XX...
 7 ...XX...
 8 ...XX...
 9 ...XX...
10 N:98
11 ...XX...
12 ...XX...
13 ...XX...
14 ...XX...
15 X..XX..X
16 .X.XX.X.
17 ..XXXX..
18 ...XX...
19 TS:#97 #98
20 W:50
```

In the preceding example, the PILOT ASCII characters 97 and 98 have been redefined to appear as the "up" and "down" arrows respectively. Since 97 corresponds to the ASCII character "a" and 98 corresponds to the ASCII character "b", the new characters are displayed every time the user presses the "a" or "b" keys. Subsequently, any TYPE instruction which displays the "a" or "b" characters will now display the new characters on the screen.

Line 19 demonstrates another way to display characters on the screen. A TYPE instruction (T:) followed by a number sign and the numeric representation of an ASCII character displays that character on the screen.

Sprites

PILOT takes full advantage of the Commodore 64's unique ability to display movable objects called sprites. Two PILOT instructions allow you to easily define and control sprites: The BIT-PATTERN instruction (B:) and the SPRITE (S:) instruction. The BIT-PATTERN instruction consists of an opcode and a sprite number, followed by the sprite definition. The sprite definition consists of 21 rows of 24 dots. The following program demonstrates how easy it is to create sprites using PILOT on the Commodore 64:

```
 1 B:0
 4 ........................
 5 ........................
 6 ........XXXXXXXXX.......
 7 ......XXXXXXXXXXXXX.....
 8 .....XXXXXXXXXXXXXXX....
 9 ....XXXXXXXXXXXXXXXXX...
10 ...XXXX..XXXXXXXXXXXXX..
11 ...XXX.X..XXXXXXXXXXXX..
12 .XXXXXX..XXXXXXXXXXXXXX.
13 ...XXXXXXXXXXXXXXXXXXXX.
14 ...XXXXXXXXXXXXXXXXXXXX.
15 ...XXXXXXXXXXXXXXXXXXXX.
16 ....XXXXXXXXXXXXXXXXXX..
17 .....XXXXXXXXXXXXXXXXX..
18 ......XXXXXXXXXXXXXXX...
19 .......XXXXXXXXXXXXX....
20 ........XXXXXXXXXXX.....
21 ....XXXX...XXXX..XXXX...
22 ...XXXXXXXX...XXXXXXXX..
23 ...XXXXXXX....XXXXXXX...
24 ........................
26 B:1
29 ........................
30 ........................
31 .........XXXXXXXXX......
32 .......XXXXXXXXXXXXXX...
33 ......XXXXXXXXXXXXXXXX..
34 ....XXX..XXXXXXXXXXXXX..
35 ....XX.X..XXXXXXXXXXXXX.
36 .XXXXXX..XXXXXXXXXXXXXX.
37 ...XXXXXXXXXXXXXXXXXXXX.
38 ...XXXXXXXXXXXXXXXXXXXX.
39 ...XXXXXXXXXXXXXXXXXXXX.
40 ...XXXXXXXXXXXXXXXXXXXX.
41 ....XXXXXXXXXXXXXXXXXX..
42 ....XXXXXXXXXXXXXXXXXX..
43 .....XXXXXXXXXXXXXXXX...
44 ...XX.XXXXXXXXXXXXX..XX.
45 ..XXXXX.XXXXXXXXXX.XXXX.
46 ..XXXXX..XXXXXX..XXXXXX.
47 ...XXXXXX.......XXXXXX..
48 ....XXXXX.......XXXX....
49 ......XX................
```

As with the NEWCHAR instruction, the ON dots are coded with an X and the OFF dots are coded with a period. To control sprites on the screen, the following SPRITE instruction is used:

S:number;operation list

where "number" is the sprite number (0-7).

Several operations that can be included in the operation list are as follows:

E1	Enable (turn on)
E0	Disable (turn off)
M1	Multi-color mode
M0	Single-color mode

X1	Double X size
X0	Single X size
Y1	Double Y size
Y0	Single Y size
P1	Priority on (sprite will be displayed behind any background data)
P0	Priority off (sprite will be displayed in front of any background data)
Cn	Set foreground color of sprite to n
Lx,y	Location (move upper left hand corner of sprite to location (x,y))
Rn	Set R color to n (only for multi-color sprites)
Qn	Set Q color to n (only for multi-color sprites)

Using the above sprite commands, the following instructions will display the previously defined sprites on the screen:

```
51 G:B0;X0;C1;E
52 S:0;C7;M0;X0;Y0;L100,50;E1
53 S:1;C7;M0;X0;Y0;L148,50;E1
```

Line 51 uses the GRAPHICS instruction to change the background and border colors to black, the cursor color to white, and erases the screen. The SPRITE instruction on line 52 specifies sprite zero, sets the color to yellow (C7), uses single-color mode (M0) with no expansion (X0;Y0), sets the (X,Y) location to (100,50) and turns the sprite ON (E1). Line 53 displays the other sprite at (X,Y) location (148,50).

Sprite animation is created by switching back and forth between sprites and/or altering the (X,Y) locations while the sprite is ON.

Multi-colored sprites are handled somewhat differently than single-colored sprites. Each sprite can have three colors. There are two commands required to set the additional colors: "R" and "Q". The color information for the sprite is coded in the bit-pattern. Each row of pixels is now treated as 12 pairs of dots. These pairs are required to generate the four possible combinations of color data:

..	To select the background color
X.	To select the "C" color
.X	To select the "Q" color
XX	To select the "R" color

The following instructions create a multi-colored sprite:

```
 1 B:0
 2 ........................
 3 ........................
 4 .........X.X.X.X.X......
 5 .......X.X.X.X.X.X.X....
 6 .......X.X.X.X.X.X.X....
 7 .....X.X.X.X.X.X.X.X....
 8 .....XX.X..X.X.X.X.X.X..
 9 .....XXXX..X.X.X.X.X.X..
10 .X.X.XX.X..X.X.X.X.X.X..
11 ...X.X.X.X.X.X.X.X.X.X..
12 ...X.X.X.X.X.X.X.X.X.X..
13 ...X.X.X.X.X.X.X.X.X.X..
14 ...X.X.X.X.X.X.X.X.X.X..
15 .....X.X.X.X.X.X.X.X.X..
16 .......X.X.X.X.X.X.X....
17 .......X.X.X.X.X.X.X....
18 .........X.X.X.X.X......
19 ....X.X....X.X.XX.X.X...
20 ....X.X.X.X...X.X.X.X...
21 ..X.X.X.X.....X.X.X.X...
22 ........................
23 G:B0;X0;C1;E
24 S:0;M1;C1;R0;Q7;L100,50;E1
```

Lines 1 through 22 define the multi-colored bit-pattern. The SPRITE instruction on line 24 specifies multi-colored mode (M1), and sets the Q color to yellow, the C color to white and the R color to black. This means that the dot pairs coded as "X." produce white, ".X" produces yellow, and "XX" produces black. The dot pair ".." becomes transparent and thus lets the background color come through. The above sprite will have a yellow body, white feet and a black and white eye area.

Although multi-colored sprites provide diversity in color, the horizontal resolution is cut in half. This is because a multi-colored sprite now takes two pixels to represent one unit of color. Another method of achieving multi-colored displays is to combine two or more single-colored sprites. In this manner, large and colorful graphic displays can be created.

Sound

PILOT on the Commodore 64 allows the programmer full access to the 6581 Sound Interface Device (SID) through use of the VOICE (V:) instruction. This feature is reserved for the more technically oriented programmer because it requires an understanding of the bits and bytes of the SID chip. But understanding the SID chip should not be too difficult since a full description (including music note values, charts, and a register map) is provided in the manual. Also included in the manual is a sample "sound editor" program, which allows you to experiment with various sounds for use in your PILOT programs. All the features of the SID chip are available to you, including ring modulation, synchronization and filtration.

Sound is created by POKing values to the SID chip through use of the VOICE (V:) instruction. The format of the voice instruction is:

V:register number,value

where "register number" is the SID chip register number and "value" is the decimal value to be stored in the register.

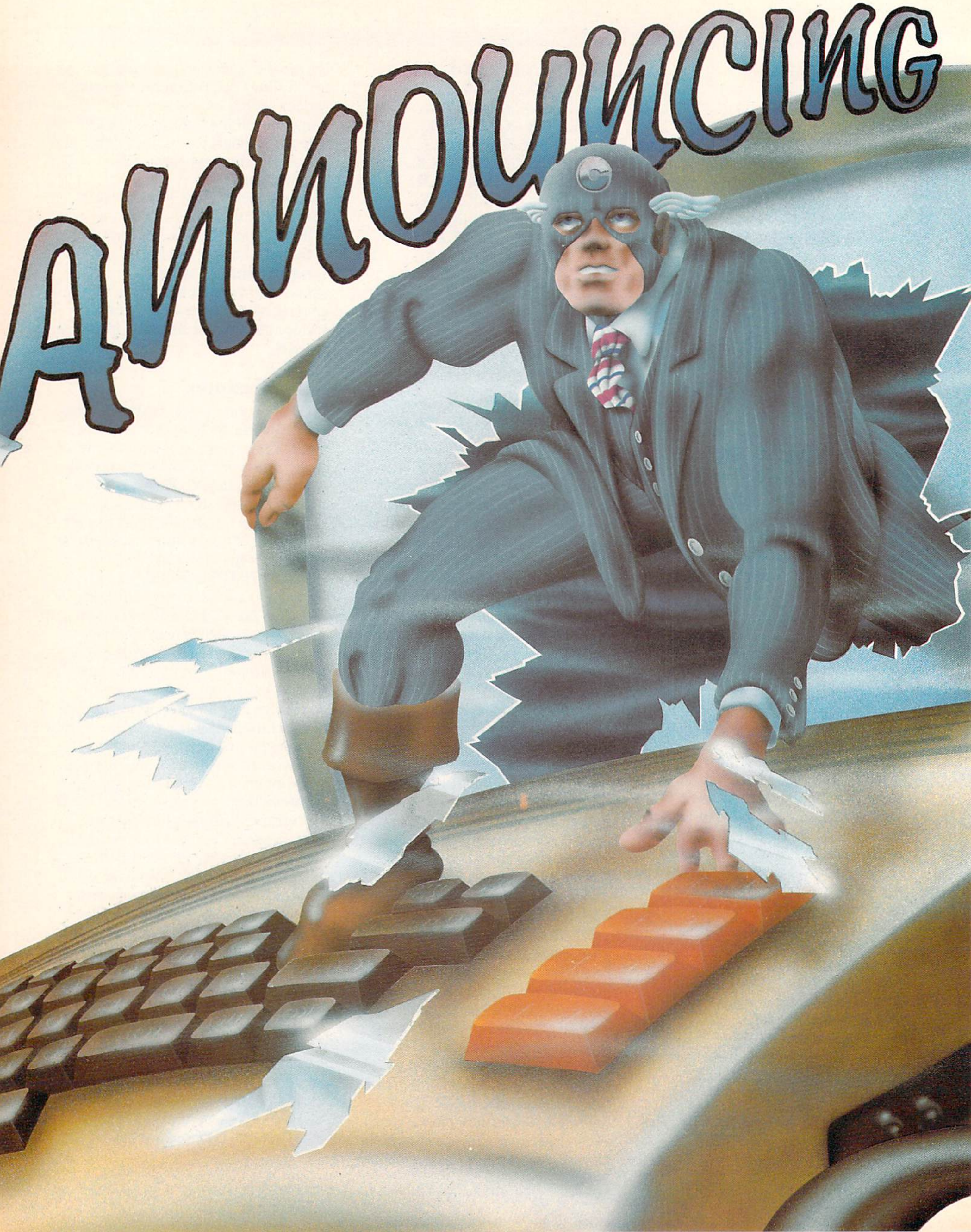
Other Features

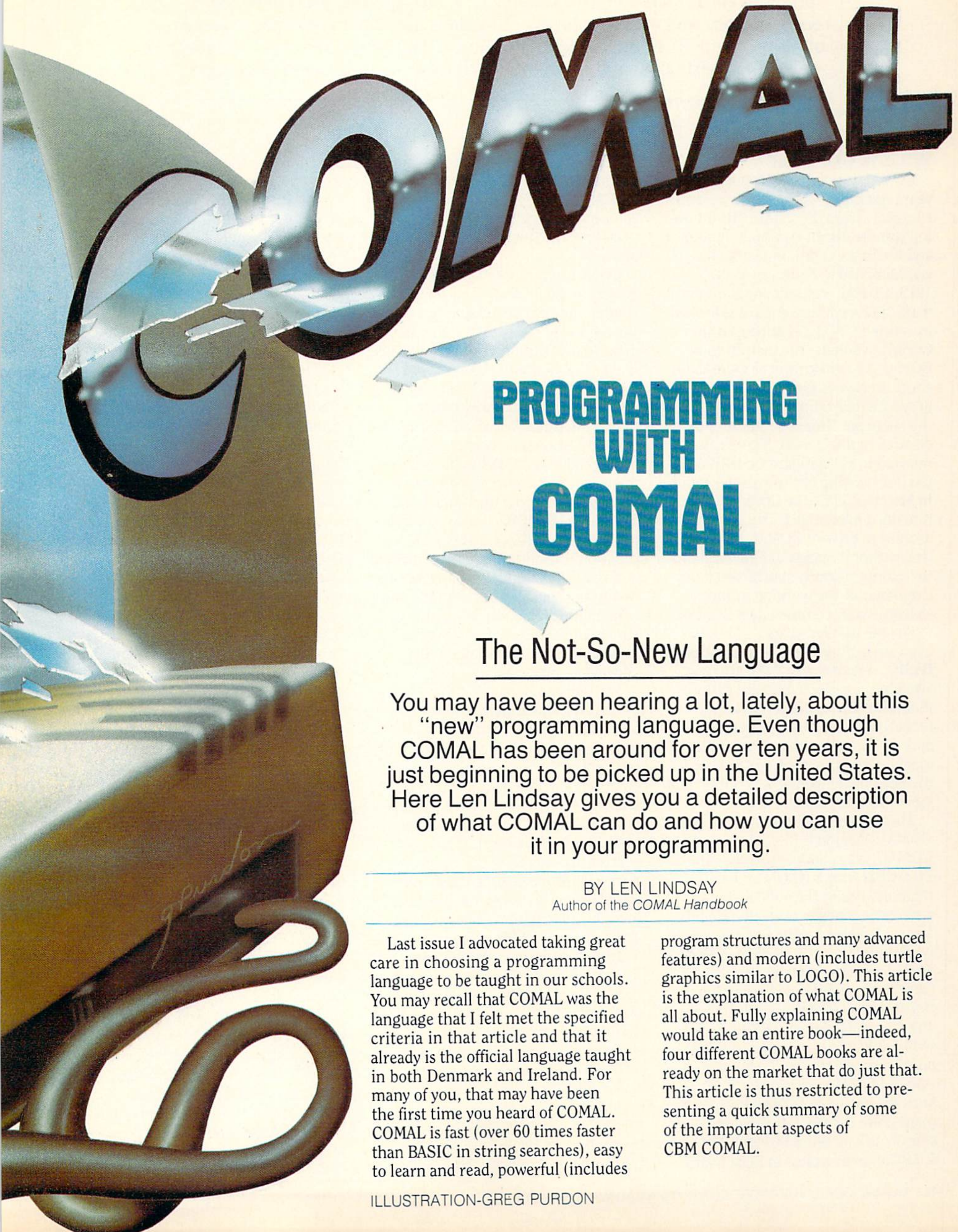
PILOT has other powerful features that we would like to mention here. The CALL (C:) instruction is used to integrate your own machine language subroutines into the PILOT system. This instruction executes a machine language JSR to the specified area of your code. A list of free memory locations is provided in the PILOT manual.

The EXTENDED VOICE (VX:) instruction is similar to the VOICE instruction except that it allows you to POKE values anywhere in memory. You can use the VX: instruction in conjunction with your machine language subroutines. Theoretically, it can also be used for direct programming of I/O chips such as the VIC chip. (This of course would be at your own risk!)

There are other PILOT instructions to maintain disk records and link segmented PILOT programs. There is also an "escape" feature, which lets you swap in special purpose routines. Also, there is a complete set of PILOT functions including random number, absolute value, string conversion, etc.

This article has touched on the major features of PILOT in an attempt to briefly describe the capabilities of the PILOT language. PILOT is simple enough for the child to use, yet sophisticated enough to satisfy the needs of the experienced computer specialist. This language provides the educational developer with unlimited creative possibilities. We can expect to see a vast array of quality educational software as a result of the implementation of PILOT on the Commodore 64. **C**


ANNOUNCING



PROGRAMMING WITH COMAL

The Not-So-New Language

You may have been hearing a lot, lately, about this "new" programming language. Even though COMAL has been around for over ten years, it is just beginning to be picked up in the United States. Here Len Lindsay gives you a detailed description of what COMAL can do and how you can use it in your programming.

BY LEN LINDSAY
Author of the *COMAL Handbook*

Last issue I advocated taking great care in choosing a programming language to be taught in our schools. You may recall that COMAL was the language that I felt met the specified criteria in that article and that it already is the official language taught in both Denmark and Ireland. For many of you, that may have been the first time you heard of COMAL. COMAL is fast (over 60 times faster than BASIC in string searches), easy to learn and read, powerful (includes program structures and many advanced features) and modern (includes turtle graphics similar to LOGO). This article is the explanation of what COMAL is all about. Fully explaining COMAL would take an entire book—indeed, four different COMAL books are already on the market that do just that. This article is thus restricted to presenting a quick summary of some of the important aspects of CBM COMAL.

ILLUSTRATION-GREG PURDON

You can call a procedure or function at any time, anywhere in your program, simply by using an EXEC statement or function call. When a running program encounters an EXEC statement, it executes that procedure before continuing on. And the procedures or functions now don't even have to be part of the program. They can be external. Thus, CBM COMAL version 2.00 more or less has a virtual memory system, just like the big mainframe computers.

What is COMAL?

COMAL is not a new language. Ten years ago Borge Christensen, a Danish educator, created COMAL. The following year the design was carefully studied, and the first COMAL implementation was launched in February of 1975. By 1979, COMAL was very popular in Denmark, and a working group was formed to refine COMAL, resulting in the COMAL Nucleus, the definition of COMAL-80, still known as COMAL for short. In May of 1982, the working group further refined and expanded the language. This redefinition was detailed in the COMAL Kernal, also referred to as "standard COMAL". At the last meeting of the working group in March of 1983, the COMAL Kernal remained unchanged. The group also decided to present COMAL to the ISO (International Standards Organization) to become officially standardized. As a member of the working group, I welcome your comments on COMAL.

For the past ten years COMAL has been struggling to replace its ancestor, BASIC. Versions of COMAL are now available on several different systems, including Commodore. Now that COMAL is available for the Commodore 64 computer, it has a good chance to take a giant leap forward in popularity. Read the following description and see if you don't agree.

There are several versions of CBM COMAL. Version 0.12, Introductory COMAL, is available on disk for any PET/CBM with 4.0 ROM and 32K memory. It is in the public domain and replaces obsolete version 0.11. It also is available for the Commodore 64 on disk or tape. The complete COMAL Kernal is implemented in version 1.02 (except for user-defined string functions). It also is in the public domain and is available for any 96K PET/CBM with 4.0 ROM, or as a plug-in ROM board. Finally, the most advanced version of COMAL is version 2.00. It contains the complete COMAL Kernal plus many enhancements. It should be available on disk for any 96K PET/CBM with 4.0 ROM, or as a plug-in ROM board. It also is available as a cartridge for the Commodore 64. Both Commodore 64 COMAL implementations include additional commands and functions to control the special graphics features of the 64.

COMAL as described in this article is version 2.00. Other versions will be similar, but may be missing some of the advanced features. A complete detailed description of all the versions is available in the *COMAL Handbook.* A list of COMAL resources and suppliers is listed at the end of this article.

COMAL is not just an advanced form of BASIC. It retains the easy-to-learn user friendliness of BASIC, but removes many of its deficiencies. Full screen editing, commonly identified with Commodore BASIC, is also available in COMAL, making it easy to edit a program. In addition, COMAL includes automatic line numbering and renumbering as a standard feature. Line numbers, however, are not relevant to a running COMAL program, but are provided for your reference while editing the program.

You will appreciate the immediate syntax check of every line as it is entered. COMAL will not accept a program line until it is correct. If an error is detected in a line you enter, a very complete and understandable error message is displayed immediately below the line, *and* the cursor is placed at the point of the error. Once you correct the line, the error message is removed from the screen, *and* whatever it previously overwrote is replaced, providing you with non-destructive error messages. This is a long overdue feature. It guarantees that every line in your program is syntactically correct. The error message does not remain on the screen after the error is corrected. And no information on the screen is lost when the error message is displayed.

To make program editing even easier, COMAL version 2.00 also includes an advanced version of the FIND command. You tell COMAL what text to find, and it will list each line that contains it, one at a time, placing the cursor on the text. You then may change or edit the line if you wish. Simply hit RETURN and the FIND continues, automatically listing the next line containing the text requested. This could even make programming fun!

CBM COMAL corrects many other peculiarities of CBM BASIC. A prime example is that now you can enter a program using BOTH upper and lower case letters. You may have wondered why BASIC couldn't recognize a program using shifted letters. It wasn't the computer's fault! Blame BASIC. But this frustrating situation is no longer a problem. In addition, you now have the option of listing your program in all upper or all lower case, or a combination. The standard COMAL listing now uses a combination, with COMAL keywords (like FOR, OPEN, and READ) listed in upper case and identifiers (variable names) listed in lower case. You will be surprised how easy it is to read a program using this combination.

The readability of a COMAL program listing is further enhanced by the automatic indentation of structures (these structures are explained below—all versions of CBM COMAL include these structures, even version 0.12). This makes it easy to see the program organization. In addition, COMAL allows you to use up to 78 characters for your variable names, all of them significant. And you don't have the restriction of keywords not being allowed as part of an identifier. BASIC will not allow a variable called FORTUNATE, since it includes the keyword FOR. COMAL not only allows it but can distinguish it from a variable called FORTUNES. As an even further aid to keeping your programs readable, COMAL allows more than just alphanumeric characters in variable names. You can also use an apostrophe ('), square brackets ([ ]), backslash (\), and back arrow (<-), which is converted to an underline character. The name LINE'COUNT, for instance, is easier to read than LINECOUNT.

Structures

COMAL includes the powerful program structures similar to those popularized by PASCAL. Once you use these structures you will wonder how you ever programmed without them. These multi-line structures are grouped as follows:

Conditional Statement Execution
IF... THEN... ELIF... ELSE... ENDIF
CASE... OF... WHEN... OTHERWISE... ENDCASE

Conditional Loops
REPEAT... UNTIL
WHILE... DO... ENDWHILE
LOOP... EXIT WHEN... ENDLOOP (not available in version 0.12)

Fixed Loop
FOR... TO... STEP... DO... NEXT (ENDFOR)

Modules
FUNC... REF... CLOSED... IMPORT... RETURN... ENDFUNC
PROC... REF... CLOSED... IMPORT... ENDPROC... EXEC

IF Structure

CBM BASIC has a very simple form of IF structure. COMAL expands it to multi-lines, including an ELSE section. Now you can provide a condition for evaluation, and if it is true, the first set of statements will be executed, but if it is false only the second set (after the word ELSE) is executed. This is illustrated below:

```
IF <condition> THEN
   True Block of Statements

ELSE                              ┌
   False Block of Statements      │ optional
ENDIF                             │ section
                                  └
```

For example:

```
IF error'count>0 THEN
   PRINT "You completed this section
   with";error'count;"errors."
   PRINT "These errors indicate areas
   needing extra practice."
ELSE
   PRINT "Fantastic, no errors in this
   section!"
   PRINT "Not many others have
   achieved this goal."
ENDIF
```

The indentation of the statement blocks is automatic, and you can see how it helps illustrate the structure. Structures can be nested, and the indentation will indicate the level of nesting. The IF structure also provides a way to include several condition checks within the same structure using the keyword ELIF (stands for ELSE IF). You may include as many ELIF sections as you wish, or none at all. The ELSE section may also be included or left out. The complete IF structure is illustrated below with an example:

```
IF <condition-1> THEN
   condition-1 True Statements

ELIF <condition-2> THEN           ┌
   condition-2 True Statements    │ optional
                                  │ additional
ELIF <condition-n> THEN           │ conditional
   condition-n True Statements    │ sections
                                  └
ELSE                              ┌
   False Statements               │ optional
ENDIF                             │ section
                                  └
```

For example:

```
IF register<bank'balance THEN
   PRINT "Your bank balance shows
   that you have"
   PRINT "more money in the bank
   than you register"
ELIF register>bank'balance THEN
   PRINT "You register more money
   than the"
   PRINT "bank balance indicates"
ELSE
   PRINT "Your register balances with
   the bank"
ENDIF
```

CASE Structure

A multiple choice type decision is possible in COMAL with the CASE structure. It allows you to present several sets of statements (each set referred to as a "case") for possible execution. The structure begins with an expression (either numeric or string) to be evaluated. Its value is compared to the expressions listed at the beginning of each case. If any expression matches, the statements in that case are executed, and the remainder of the structure is skipped. If no match is found in any of the cases, the OTHERWISE statements are executed. This structure replaces the primitive ON X GOTO <line1>, <line2>,<line3>...

```
CASE <expression> OF
WHEN <expression list-1>
  Statements-1
WHEN <expression list-2>          optional
  Statements-2                    additional
                                  case
WHEN <expression list-n>          sections
  Statements-n
OTHERWISE                         optional
  Otherwise Statements            section
ENDCASE
```

For example:

```
CASE choice$ OF
WHEN "H", "h", "?"
  EXEC instructions
WHEN "A", "a"
  CHAIN "ADD"
WHEN "S", "s"
  CHAIN "SUBTRACT"
OTHERWISE
  PRINT "I don't understand your
  choice."
  PRINT "Your choices are:
  PRINT "A—Add"
  PRINT "S—Subtract"
  PRINT "H—Help—instructions"
ENDCASE
```

(Version 0.12 does not include this structure.)

Conditional Looping Structures

Most programs have statements that are to be executed over and over. There are three ways to conditionally exit such a loop: prior to beginning the loop, after completion of the loop, or somewhere within the loop. COMAL has a specific structure that allows for each.

The REPEAT loop executes the loop statements first, and then checks the condition specified in the UNTIL statement. The loop is repeated until the condition evaluates to TRUE.

The WHILE loop checks a condition first. If it evaluates to FALSE, the loop is considered finished and the statements are skipped. If it evaluates to TRUE, the statements are executed, and then the condition is checked again.

The loop structure continues executing its statements as long as the EXIT WHEN condition is FALSE.

```
REPEAT
  Statements
UNTIL <condition>

WHILE <condition> DO
  Statements
ENDWHILE

LOOP
  Statements
EXIT WHEN <condition>
  Statements
ENDLOOP
```

For example:

```
REPEAT
  INPUT "What is the answer": reply
  tries:+1
UNTIL reply=answer OR tries>2

WHILE NOT EOF(in'file)
  READ FILE in'file: text$
  PRINT text$
ENDWHILE

LOOP
  READ FILE in'file: name$
EXIT WHEN name$="*END*"
  pointer:+1
  array$(pointer):=name$
ENDLOOP
```

Fixed Loop Structure

Often you will want to execute a block of statements a specific number of times. COMAL has a structure similar to the FOR...NEXT of BASIC. However, COMAL converts the keyword NEXT into the keyword ENDFOR to be compatible with the terminating statement of other loop structures. COMAL will still accept the keyword NEXT if you type it in, but it then will convert it to ENDFOR for you.

```
FOR <controlvar>:=<start> TO
<end> STEP <step> DO
  Statements
ENDFOR <controlvar>
```

The "STEP <step>" section of the FOR statement is optional. If omitted, a step of 1 is used, just as with BASIC. Also similar to BASIC, COMAL allows you to use just ENDFOR without having to specify the <controlvar>. However, it goes a step further and inserts the <controlvar> for you, improving your program listing. Note that the <controlvar> is compared to the <end> value before the statements are executed. Thus it is possible to skip the FOR loop statements entirely. As an example, the following loop statements would be skipped (not executed even once):

```
starting:=1; ending:=0
FOR temp:=starting TO ending
  PRINT "Now inside the FOR loop"
ENDFOR temp
```

An example of a loop to read 12 values from data statements and store them into a string array follows:

```
FOR month:=1 TO 12 DO
  READ month'name$(month)
  PRINT "Month number";month;"is
  named";month'name$(month)
ENDFOR months
```

COMAL also allows a quick one-line FOR statement without the use of the ENDFOR terminator.

```
FOR <controlvar>:=<start> TO
<end> STEP <step> DO <statement>
```

A common example is a pause loop, shown below (this example also illustrates the use of the NULL statement, which, as its name implies, does absolutely nothing):

```
FOR pause:=1 to 500 DO NULL
```

Procedures and Functions

Multi-line procedures and functions are one of COMAL's specialties. They can be easily used by the beginner, yet offer options that should satisfy even the most advanced programmers.

It is easy to write modular programs in COMAL. You can call a procedure or function at any time, anywhere in your program, simply by using an EXEC statement or function call. When a running program encounters an EXEC statement, it executes that procedure before continuing on. Procedures are called by name (remember, line numbers are not relevant to a running COMAL program). The procedures and functions allow parameter passing as well as both local and global variables. And the procedures or functions now don't even have to be part of the program. They can be external. Thus, CBM COMAL version 2.00 more or less has a virtual memory system, just like the big mainframe computers. Procedures can be nested, and one program can call another, or even can call itself (recursion). COMAL allows user-defined numeric, integer and string functions. The value of the function is returned via a RETURN statement (different from the RETURN statement of BASIC). Although I will concentrate on describing procedures in this article, functions share all their flexibility, except are called with a function call and return a value, instead of being called with an EXEC statement.

The most important line in a procedure is the first line, which I will refer to as the header. This is where you give the procedure a name, indicate the parameters (if any), and specify whether it will be a CLOSED procedure (with local variables) or not. Version 2.00 extends the header with an optional external indicator. Thus the PROC statement can be rather complex. This provides you with flexibility. You don't have to use all the options, bells and whistles. You can start with a basic procedure.

The simplest PROC statement only gives the procedure a name. All variables remain global, and parameters are not used:

```
PROC <procedure name>
```

```
PROC request
```

You can make all the variables within the procedure local by adding the word CLOSED to the end of the PROC header line. A CLOSED procedure does not know about any of the variables used in the main program, and the program similarly does not know about the variables inside the procedure.

```
PROC <procedure name> CLOSED
```

```
PROC pause CLOSED
```

However, even a CLOSED procedure can share values from the main program via parameter passing or the IMPORT statement. The variables listed in the parameter list for the procedure are assigned initial values supplied by the calling EXEC statement. Simple value passing is only "one way"—into the procedure. The PROC header looks like this:

```
PROC <procedure name>(<parameter
list>) CLOSED
```

```
PROC label'print(name$,address$,
city$,state$,zip) CLOSED
```

COMAL also allows you to pass values back to the calling statement. Simply include the keyword REF before any parameter you wish to be a "two way" parameter. That variable will then be called by "reference", and used as an alias for the matching variable in the calling EXEC statement. The variable name is still considered local and won't conflict with any variables with the same name in the main program. And the variable name called by reference does not have to be the same as the one in the calling statement; in fact, it usually will be different. As an example, let's illustrate how to input some text, and then convert any characters that are not a digit into a period and each digit into the letter D:

```
DIM reply$ OF 80
INPUT "Enter some text including dig-
its:": reply$
EXEC convert(reply$)
PRINT "Converted it now looks like
this:";reply$
PROC convert(REF text$) CLOSED
  FOR temp:=1 TO LEN(text$) DO
    IF text$(temp) IN "0123456789"
    THEN
      text$(temp):="D"
    ELSE
      text$(temp):="."
    ENDIF
  ENDFOR temp
ENDPROC convert
```

Now here is a sample run of this program:

```
RUN
```

```
Enter some text including digits:
ABC123XYZ5
Converted it now looks like this:
...DDD...D
```

In addition to illustrating the use of two-way parameter passing, the example shows how a part of a string can be changed without affecting the rest of the string. You also can see how automatic structure indenting increases program readability.

A CLOSED procedure can also share variables with the main program via the IMPORT statement. The IMPORT statement is actually part of the procedure. A list of variable names is included, all of which will be shared with the main program:

```
IMPORT name$,score
```

Version 2.00 takes all the power and flexibility of procedures and functions and adds the capability of allowing them to be external to the calling program. This means that you can have all of your most used procedures on one disk. As long as this disk is in the drive, your programs can use any of the procedures on that disk at any time by simply including only the PROC header, ending with the word EXTERNAL and the correct file name:

```
PROC <procedure name>(<parameter list>) CLOSED EXTERNAL
<file name>

PROC quicksort(left,right, REF
names$()) CLOSED EXTERNAL
"quicksort.e"
```

An external procedure is retrieved from disk only when it is needed. And as soon as it is finished executing, it is removed from memory. Thus external procedures in effect give you a virtual memory system. Plus, now when you update your procedure, you only need to update it once on your master procedure library disk. Then every program calling it will be using the most up-to-date version of each procedure. And finally, your programs will be much smaller, now that the procedures do not need to be redundantly coded in every program that uses them. This will allow more programs to be stored on a disk.

With external procedures and functions, program chaining, machine code linking, and advanced error handling, COMAL has progressed beyond a language only for beginners. Beginners will still find it most enjoyable, and later will be happy to learn about all the advanced features.

Some Other Advanced Features

Some of the advanced features now available in COMAL include things taken from PASCAL and ADA, as well as things unique to COMAL. A whole array can be written to disk with one statement. Later an array of similar dimensions can be initialized with that disk file, also using only one statement. Sequential files can be read and written in two different ways. One method, of course, is compatible with files written by BASIC, allowing you to access files from BASIC programs you already have. The other method uses a character count for text records, allowing the text itself to contain any of the ASCII characters, including those usually restricted by BASIC's method.

COMAL now includes a "time and date stamping" feature that is very useful. Each time a file is stored on disk, it includes the system time and date. Thus if you have three different versions of a program on one disk, but don't remember which is the latest version, you now can check the time and date stamp on each file, and easily determine which is the latest version. An enhanced directory listing can now also include the time and date that each file was stored on the disk. Backup programs can also use this information, to backup only the files that have been added or changed within the last week, making a weekly backup much easier, as well as requiring less time and fewer disks.

The advanced INPUT facility that COMAL now includes is remarkable. It provides complete protection from having a user messing up a formatted screen during input, without your having to write complex procedures yourself to control the input. The INPUT statement now does not accept characters that obviously should not be included in any input request. Thus a cursor up or cursor down is ignored. Home cursor has been redefined to mean "put the cursor back to the first position of the input field". Clear screen has been redefined to do the same as home cursor, but also erases all the previous input in that field, giving you a fresh start. COMAL also will not allow a user to go into quote mode or insert mode during an INPUT request. Reverse field is ignored as well.

If you think this is fantastic, COMAL expands it even further with the INPUT AT statement. This statement allows you to specify at what position on the screen the input is to start *and* how many characters at most will be accepted. Thus with this one statement you can specify that the input is to start at row 10 position 5 and cannot be more than 6 characters long. Even a beginner can now produce some really professional bullet-proof programs without too much effort.

In addition to advanced INPUT features, COMAL also provides some advanced OUTPUT features. It includes PRINT USING, which unfortunately was never implemented in CBM BASIC. It also allows easy switching from output to the screen to output to the printer (output can also be directed to a disk file). TAB and ZONE work the same on both screen and printer (not so in CBM BASIC). And if you have an ASCII printer (any non-Commodore brand) you may already realize that PET ASCII is not quite the same as standard ASCII. COMAL includes an option to automatically convert PET ASCII to standard ASCII on any printed output. COMAL also allows the option of sending a line feed with every carriage return. Some printers need it while others can't handle it.

Value assignment to variables is more advanced than CBM BASIC. COMAL precedes the equal sign in assignments with a colon (:=) as does PASCAL. This makes it easier to distinguish between an assignment and a comparison. However, you do not have to type in the preceding colon; COMAL will add it for you. COMAL allows incrementing and decrementing a variable the same way as BASIC, but also includes an ALGOL-like method. Just precede the plus or minus sign with a colon and it becomes an increment or decrement symbol, consistent with the colon before the equal sign. For example, here is how to subtract your money bet from your total money:

```
BASIC: T=T-B

COMAL: TOTAL:-BET
```

String concatenation is also possible using this shortcut:

```
BASIC: T$=T$+R$

COMAL: TEXT$:+REPLY$
```

COMAL also includes both TRUE and FALSE as system constants. FALSE is always equal to 0 and TRUE equal to 1. When you are using a TRUE/FALSE comparison, it is much easier to follow and is less ambiguous if you use the word TRUE or FALSE rather than the numbers 0 or 1. COMAL also includes some system variables that are very useful. EOD is set to TRUE when the last data item is read from data statements. EOF is set to TRUE when the end of file is encountered in an input data file. COMAL also includes a statement to disable the stop key and a system variable, ESC, that then is set to TRUE while the STOP key is depressed. Your program can then use the STOP key for any function you wish.

In addition, COMAL provides an advanced error handling system that you can use in your programs. Now, if your program encounters a run-time error (like "file not found"), it can take appropriate actions rather than just stopping with an error message.

The concept of a "package" from ADA has also now been included in COMAL. A package is a self-contained system complete with initialization and its own procedures and functions, which can be global or private. A package is an advanced technique, taking the external procedure one step further. Most users will not create their own packages, but you may find many application programs using them, or even software houses selling packages, just as they now market tool kits for BASIC.

Finally, if you are using a Commodore 64, both versions 0.12 and 2.00 have been specially adapted to control the graphics features of the computer directly from COMAL (no more PEEKs and POKEs). This includes high resolution graphics, turtle graphics (similar to those in LOGO), and sprite control commands. Version 2.00 also includes similar commands to control the sound synthesizer features. These features alone should lead the way for COMAL to replace BASIC. C

Quick Buyer's Guide to COMAL

COMAL Interpreters

All versions of CBM COMAL were written by UniComal Denmark ApS, formerly part of Instrutek. Both CBM COMAL versions 0.12 and 1.02 are in the public domain. If you don't know someone who can make you a copy, they are available on the *COMAL Handbook* disk for $15 from Reston Publishing in Reston, Virginia, or the COMAL Interest Group in Madison, Wisconsin. (See below for addresses of these and other resources.)

The plug-in COMAL ROM board is available from Instrutek or the COMAL Interest Group. It can be either standard (version 1.02) or deluxe (version 2.00).

Distribution arrangements for CBM COMAL version 2.00 and the special adaptations for the Commodore 64 computer had not yet been finalized when this article was written. Hopefully the disk versions will also be placed in the public domain and thus can be included on the *COMAL Handbook* disk mentioned above. The CBM COMAL version 2.00 cartridge for the Commodore 64 may be available from Commodore some time in the future.

COMAL Books

**COMAL Handbook* by Len Lindsay, Reston Publishing.
**Beginning COMAL* by Borge Christensen, Ellis Horwood Publishing.
**Structured Programming With COMAL* by Roy Atherton, Ellis Horwood Publishing.
**Foundations in Computer Studies With COMAL* by John Kelly, Educational Company of Ireland Limited.
Available soon:
**Graphics and Sound With COMAL on the Commodore 64* by Len Lindsay, Reston Publishing.

COMAL Information

**COMAL Catalyst*, newsletter, Gerald Hasty Company.
**COMAL Bulletin*, newsletter, Ellis Horwood Publishing.
**Riomhiris na Scol*, newsletter.
**Information About COMAL*, COMAL Interest Group.

Resource List

*Borge Christensen; States Training College, Ostergade 65, DK-6270 Denmark.
**COMAL Bulletin;* Editor, Roy Atherton; Computer Education Centre, Bulmershe College of Higher Education, Reading, Berkshire, U.K.
**COMAL Catalyst;* Publisher, Gerald Hasty; 5130 E. Charleston, Suite 5-315, Las Vegas, NV 89122; 702-452-3368.
**COMAL Catalyst;* Editor, Len Lindsay; 5501 Groveland Terrace, Madison, WI 53716; 608-222-4432.
*COMAL Interest Group; 505 Conklin Place, Madison, WI 53703.
*COMAL Users Group USA; 5501 Groveland Terrace, Madison, WI 53716; 608-222-4432 (include Self Addressed Stamped Envelope for reply).
*COMAL Users Group England; John Collins; 4 Grimthorpe House, Percival Street, London, EC1V OB5, U.K.
*COMAL Users Group Alberta; % Tom Garraway; Division of Educational Research Services, University of Alberta, Edmunton, Alberta, Canada T6G 2G5.
*Educational Company of Ireland Ltd.; Ballymount Road, Dublin 12 Ireland.
*Ellis Horwood Limited; Market Cross House, Cooper Street, Chichester, West Sussex, PO19 1EB, U.K.
*Instrutek; Christiansholmsgade, DK 8700, Horsens, Denmark; Phone: 05 61 1100.
*IPUG; Mick Ryan; 164 Chesterfield Drive, Riverhead, Sevenoaks, Kent, U.K.
*Reston Publishing; 11480 Sunset Hills Road, Reston, VA 22090; 703-437-8900.
**Riomhiris na Scol;* Colaiste an Spioraid Naomh, Bishopstown, Cork, Ireland.
*UniComal Denmark ApS (See Instrutek).

More Next Issue

Next issue I will explain how easy it is to control the graphics and sound on the Commodore 64 computer using the built-in commands and functions of COMAL. If you have any comments on COMAL, please let me know. If you like COMAL, please tell everyone about it, especially the schools in your area. Help spread the good news.

Random Thoughts

Random and pseudorandom numbers can be useful programming tools for many applications. This first part in our new series explains a little about what random numbers are and how to use them to your advantage.

BY MARK ZIMMERMAN

What This Series of Articles is About

"Random Thoughts" will be a series of articles about computing centered on the theme of random numbers and their uses. I will talk about how to generate random numbers, the different types of random number sequences and their applications, the physical systems that can be simulated using random numbers, and much more.

The goal of "Random Thoughts" is to give you mental tools for working with random numbers in your programs, and to suggest ideas for further development. I hope that I will be able to use "we" instead of "I" in future columns—so please write in (% Commodore Magazine) with your thoughts, questions, answers, and comments! I'll try to respond to all letters. I'll also try to include something for everyone in each article, from novice through "hacker" through computer scientist/mathematician.

Getting Started

First, if you've never done so, turn your computer on and generate some random numbers. In BASIC, try:

```
FOR I=1 TO 10: PRINT RND(1):
NEXT I
```

You should get ten numbers on the screen, all different, with no apparent pattern among them except that all lie between 0 and 1. Try executing the same statement again; you'll get 10 other numbers in the same range. What's happening?

The BASIC function RND is unlike all the other BASIC functions (SIN, SQR, EXP, etc.) in that it doesn't seem to give a predictable answer every time. (Actually, it *is* predictable; more on this later!) The RND function also has a strange dependence on its argument, the number in parentheses after it. If you say

```
PRINT RND (−1)
```

you'll get a not-at-all random result; try it a few times. Asking for RND of any negative number similarly acts oddly.

Here's the secret: with a positive argument, RND ignores the number in parentheses after it. Instead, it takes the *previous* result of a call to RND and begins to manipulate it. (If you haven't called RND before but have just powered-up the computer, there's a fixed initial value assumed for the previous result.) RND takes the bytes of the old result and shuffles them around, adds and multiplies and truncates, and creates a new result that has very little resemblance to the old one. It then stores the new number in memory (to be used the next time RND is called) and finally gives the new result to you, to do with as you please.

If you call RND with a negative argument, as in PRINT RND (−17), you don't get a random result. RND takes the negative number you give it, scrambles it up a bit, and replaces the old previous result in memory with

ILLUSTRATION—KARL KOFOED

that. Thus, giving RND something negative allows you to conveniently reset that memory space.

So, if you're going to write a game program, for example, and want the game to go differently every time you run it, it's not enough to just use RND to generate random numbers. The sequence you get will be the same every time the game is started after powering-up the computer. You have to call RND once with a negative argument when you're beginning the game, and that negative argument should be different in order to get a different game. You can ask the player to give you a number (like the date or time) or you can include a statement like

X = RND (−TI)

which takes the time (in 60ths of a second), since the machine was turned on from variable TI, makes it negative, and gives that to RND. Just don't use the value X that you get for game control; discard it, and thereafter call RND(1) to get a good random number.

Random or Pseudorandom?

Random numbers, especially in connection with computers, raise a philosophical question: Are they really random? First, though, what do we mean by "random"? For my purposes, there are two important criteria of randomness: distribution and independence. Let me explain what I mean by each in turn.

First, random numbers have to have some sort of controlled distribution or range. The RND function of BASIC gives you results in the range between 0 and 1; casting a die gives you results from the integers 1, 2, 3, 4, 5, and 6. Sometimes the distribution is not equally likely throughout the range; examples include the values you get from throwing two dice and adding the spots shown (where 7 is more likely than 11), or the distribution of money or intelligence in a population, where there are a lot of people near some average value and fewer and fewer as one moves away from that average.

The second important quality that random numbers must have is some sort of controlled independence from each other. The independence may be total, so that knowing all the previous numbers tells you nothing about the next one. Or, it may be partial independence, so that previous numbers bias the likelihood for the next number one way or another. Radioactive decay seems to be totally independent for isolated atoms; the results of tossing a coin don't seem connected to each other, either. On the other hand, if your parents are tall, you're likely to be tall too. If you are playing *Monopoly* and have a piece on Boardwalk, the next place you stop is not likely to be the same as it would be if you were starting out in jail.

Later columns in this series will discuss specific examples of random number distributions in detail. Many important distributions have names of famous mathematicians (Gauss, Poisson, etc.) attached to them; these distributions frequently have significant applications in systems of physical interest, which we'll also discuss. We'll explore how to take one distribution of random numbers (such as those produced by RND) and turn them into another distribution.

But to return to the philosophy: How can a computer, which does only exactly what you command it and which is perfectly predictable, ever produce randomness? And how can nature, which obeys well-defined physical laws, be random either?

Well, as far as scientists know today, some areas of nature really are random. Quantum mechanics, the theory of very small systems such as atoms and nuclei, seems to demand that there be an intrinsic, unavoidable uncertainty in the results of some measurements. This isn't a "classical" effect, and our lifetime instincts, developed from working with large, macroscopic objects like stars, bowling balls or grains of sand, simply don't apply to electrons or photons.

But even excluding quantum mechanical effects, there seems to be a lot of randomness in the world. The stock market, or next week's weather, or the shuffled cards in a deck seem to be more or less random and unpredictable. But if you could follow each card in each shuffle, or follow each gust of wind, or each decision to buy and sell shares, you would find that they all are simply obeying fairly simple (or not-so-simple) laws. The systems are just so complicated, contain so many parts, or depend so critically on tiny differences, that it becomes humanly impossible to keep track of all of the details. So, even systems that are deterministic and predictable in theory act randomly for all practical purposes.

A computer generates random numbers in the same way. When you call the RND function, the bits in the previous random number you got are scrambled about and manipulated in a deterministic, but complicated pattern. The outcome is another number, which has only a very subtle relationship to the last one, if the RND function is doing its job.

These numbers thus shouldn't be called random in the strictest sense of the word. They are *pseudorandom;* that is, they appear to be random, but aren't.

Pseudorandom numbers actually have some advantages over truly random numbers in computer programming. The main plus of a pseudorandom sequence is that you can recreate it, if and when necessary. If you are trying to debug a complicated program that contains "random" elements, it helps a lot to be able to change parts of it and then run *exactly* the same tests again, without having to worry about the fluctuations that a truly random sequence of numbers would introduce. Pseudorandom numbers can also be valuable in constructing codes or ciphers, for transmitting information that needs to be kept secret. The transmitter can generate a pseudorandom noise sequence and use it to cover up the message being sent, and the receiver can generate the same sequence and use it to recover the original signal.

We'll return to this question of random vs. pseudorandom in later columns of this series. A related problem is, "How can one recognize randomness?" That is, is the series: 1, 2, 3, 4,... random? It doesn't look very random, but perhaps it came up in the course of throwing a die, or perhaps in the decimal expansion of pi. Would it be random in those cases? Think about it...

Closing Hints

We'll finish this month's comments with a couple of practical hints for working with random numbers in your programs. Just because random numbers are so unpredictable, they frequently make finding program bugs difficult. In fact, many programs are never tested out thoroughly enough to see some subtle bugs involving random numbers, and thus are never really "right".

One excellent technique for debugging a program section that incorporates random variables is to *consider the extremes.* That is, imagine what will happen if RND returns a value just barely greater than 0, or just barely less than 1. For instance, to get a random integer in the range 1 through 6, one frequently sees the BASIC statement

X = 1 + INT(6*RND(1))

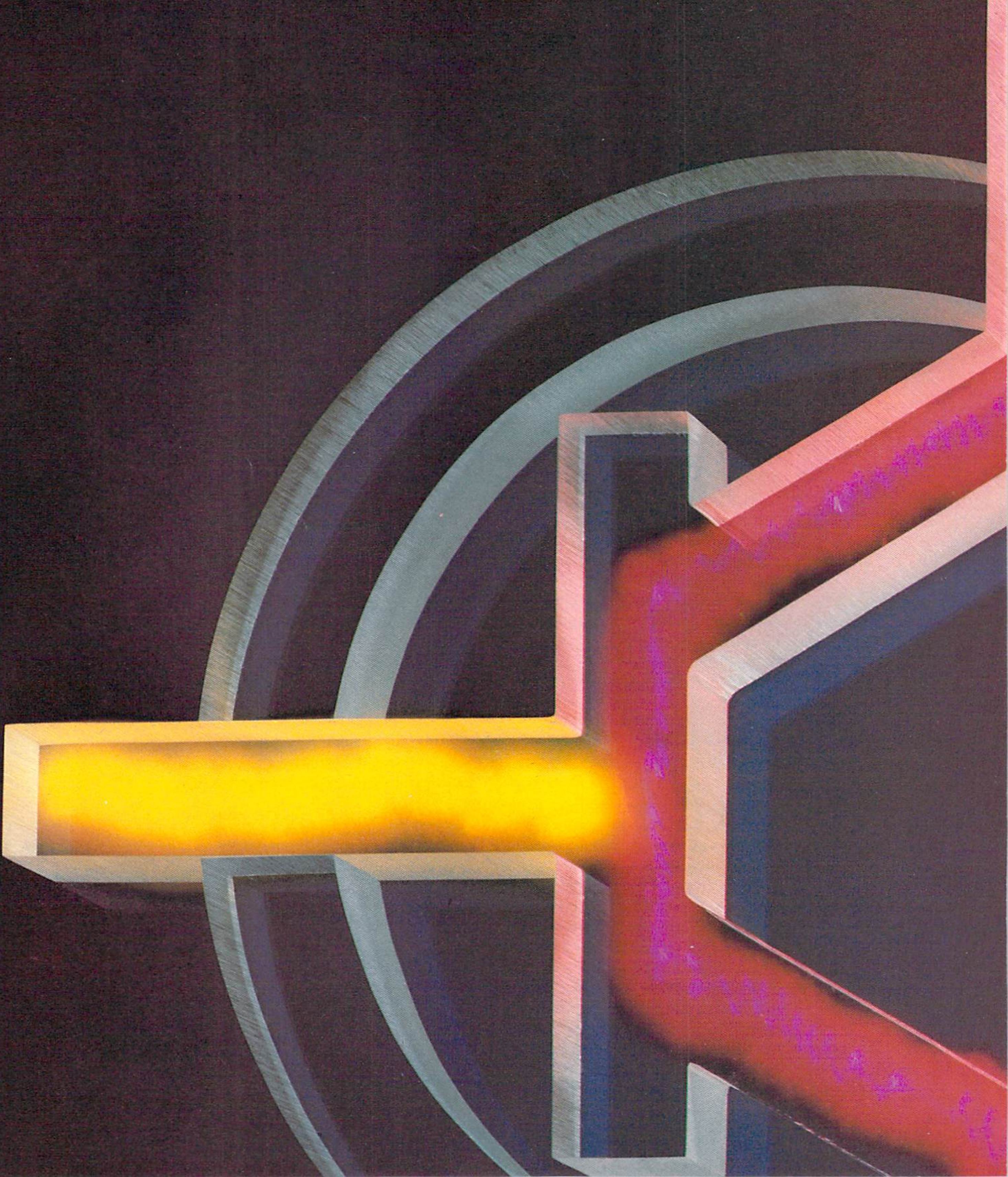
I have given this statement in its correct form. If RND(1) is barely above 0, X will be 1; if RND(1) is barely under 1, X will be 6. But in the heat of programming, it's easy to forget to add the "1" to INT(6*RND(1)). Sometimes, when dealing with integers over a range other than 1 to N, it's also easy to get the multiplier inside the INT function wrong. The "consider the extremes" test will help avoid such mistakes.

A second valuable debugging technique is to *try a smaller case.* Instead of testing out your card-shuffling technique on a 52-card deck, try it out mentally on a 2- or 3-card deck first!! This way, you can easily spot an overlooked case. In the card-shuffling example, one frequently sees excellent programs that are based on the technique of exchanging cards in an ordered deck to get the shuffled output. This is all very well and good, but when designing such a routine, don't forget to give a card a chance to end up in the same place it started! If a real deck is shuffled, there is a chance for a card to still be in its original location; there's even a tiny chance for the whole deck to end up in its original order. A shuffling program that doesn't allow that possibility is flawed.

So, there are two simple but powerful tools for checking your random-based program routines: consider the extremes, and try a smaller case. Use them!

Next time, I'll begin looking at how you can take the uniformly distributed numbers given by RND and transform them into the distribution you need. I'll also give some more examples of real-life random systems and how you can model them on your computer. **C**

Starting with Bits



and Pieces

Part 1

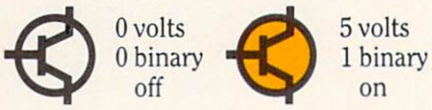
Jeff Hand, whom you may already know as the friendly (not to mention helpful) Systems Operator on the Commodore Information Network, explains the rudiments of how computers "think" and the relationships among several different number systems commonly used with micros.

BY JEFF HAND

Many times when I'm explaining computers to beginners I'm confronted with the question, "How does the computer think?" That's a pretty tough question to answer with just a short explanation, so sometimes I get frustrated trying to get my point across. I thought a series of articles might be a good format for presenting this information. It is not meant to be in any way definitive, but rather to help you get your feet wet. Well, here we go.

SCULPTURE & PHOTO—BOB EMMOTT

I'm sure most computerists know that computers "think" with zeros and ones. That's true, but how? The simplest place to start is with a transistor. Commodore computers use what is known as TTL, which stands for transistor-transistor logic. Does this mean the creation of those ones and zeros is done with mirrors? No, transistors. Here is what an engineer draws to represent a transistor:



You don't have to understand how the transistor works to understand what it does. As far as the computer is concerned the transistor has two possible states: off and on. These two states can be interpreted in a number of ways. They can be interpreted on a numerical level as a 0 (off) or 1 (on); on a switch level as switch off or switch on; or on an electronic level as 0 volts (off) or 5 volts (on). Usually "on" is represented by a shaded transistor, and "off" by a non-shaded transistor.

So far it seems simple enough, right? But you knew it wasn't going to stay that way, didn't you? We could expand on each of these concepts, but for our purposes (to understand programming) we'll stick to the numerical representations. Now, what kind of system can be built with just zeros and ones? The system is called binary or base 2. But before we go into our explanation of binary, let's backtrack to something we are more familiar with—the decimal system. For purposes of illustration I would like to represent the decimal number system as a car mileage counter.

0	0	0	0
1000's	100's	10's	1's
10^3	10^2	10^1	10^0

When you have traveled one mile the odometer will read:

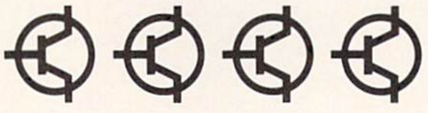
0 0 0 1

and continue 0002, 0003, 0004 up to 0009 to represent each mile. When you reach ten miles the odometer will shift to the next column:

0 0 1 0

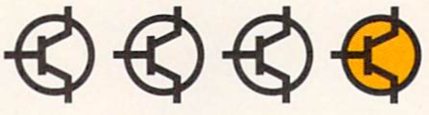
and then continue counting 0011, 0012, 0013 up to 0019.

The same kind of system that was established for the tens can also exist in a binary "odometer" system as follows (remember that each transistor represents a zero or a one):



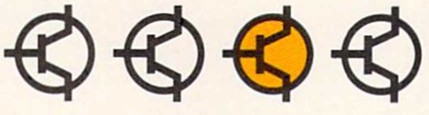
0	0	0	0
2^3	2^2	2^1	2^0

Here all transistors are off and have 0 volts. Now when we travel one mile, the binary "odometer" looks like this:



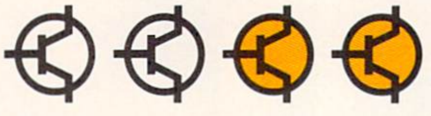
0 0 0 1

Two miles looks like this:



0 0 1 0

Three miles:



0 0 1 1

and so on.

From this you can see how computers build a number system based on electronics. We imposed the number system on the electronics and now the computer can do all its operation in a binary format. This is the bottom level in computers. From this all the other systems in computing are built.

To help you understand this concept, here is a table of number system equivalents:

Binary (for computers)	Decimal (for people)	Hexadecimal (compromise between people and computer)
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F
1 0000	16	10
1 0001	17	11

The hexadecimal system was built as a compromise between computers and people, because the binary system can get very confusing sometimes, especially when the number gets very large or small. To avoid confusion, the hexadecimal number system was created. The system uses a base of 16; 0 to 9 and A (10) through F (15). Notice the relationship between binary digit grouping and the hex numbers.

```
0001  1011  0111 (Binary)
  1     B     7  (Hexadecimal)
        or
1010  0100  1111 (Binary)
  A     4     F  (Hexadecimal)
```

Hexadecimal allows easy translation from binary so the binary is understandable to people. You can build any number system, but these are the ones used in microcomputers.

Converting Number Systems

The easiest way to see the relationship between numbers is by adding up the weighted or power values. For instance, take the decimal number 53,218:

5	3	2	1	8
10^4	10^3	10^2	10^1	10^0

It's easy to see that this decimal number is fifty-three thousand two hundred-eighteen, because we are very familiar with the decimal system and can add up the numbers ourselves quickly. But let's take a closer look at how we did it. Let's examine the weight or value of each column:

10^4 = 10,000 (ten thousands)
10^3 = 1,000 (thousands)
10^2 = 100 (hundreds)
10^1 = 10 (tens)
10^0 = 1 (units)

$53{,}218 = (5 \times 10^4) + (3 \times 10^3) + (2 \times 10^2) + (1 \times 10^1) + (8 \times 10^0)$

This weighted system can also be applied to binary and hexadecimal as follows:

1	0	1	1	1
2^4	2^3	2^2	2^1	2^0

$2^4 = 16$ $2^3 = 8$ $2^2 = 4$ $2^1 = 2$ $2^0 = 1$

To find out what this number is equivalent to in decimal, all we have to do is add up the weights:

$(1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$

$16 + 4 + 2 + 1 = 23$ (decimal)

The same procedure can be applied to a hexadecimal number:

2	4	F	A	B
16^4	16^3	16^2	16^1	16^0

$16^4 = 65{,}536$ $16^3 = 4096$ $16^2 = 256$
$16^1 = 16$ $16^0 = 1$

$(2 \times 16^4) + (4 \times 16^3) + (F \times 16^2) + (A \times 16^1) + (B \times 16^0)$

$(2 \times 65{,}536) + (4 \times 4096) + (15 \times 256) + (10 \times 16) + (11 \times 1) = 131072 + 16384 + 3840 + 160 + 11 = 151{,}467$

That's an awfully big number! But we don't usually have to worry about hexadecimal numbers of this size on the microcomputer because there is an eight-bit limitation. (The word "bit" is an abbreviation for "binary digit". For example 0001 is a grouping of four bits and 0001 1111 is eight bits. Each grouping of eight bits is called a byte.) For example, the largest number you can generate without some of the more advanced techniques I'll show you in another article is 1111 1111, which is FF in hex or 255 in decimal.

We've covered hex-to-binary, binary-to-hex, hex-to-decimal and binary-to-decimal conversions. So the only ones we've missed are decimal-to-binary and decimal-to-hex. There are several methods for doing these two conversions, but I prefer a method I'll call trial-and-error, for lack of a better term. Once you understand the relationship between the number systems you'll be able to do this quickly in your head (with some practice). The first step is to write down the decimal number. Next write out all the hex or binary weights that are less than the decimal number as follows:

53 (decimal)
32 16 8 4 2 1 (binary) or 16 1 (hexadecimal)

For the binary numbers put a 1 under each of the numbers that will contribute to adding up to a decimal number 53:

32 16 8 4 2 1
1 1 0 1 0 1 This is the binary equivalent of 53.

To find the hex number determine how many times 16 will divide evenly into 53 without exceeding it. Sixteen goes into 53 three times with five remaining. Therefore the hex number is 3 (times 16) + 5 (times 1) or 35. Try practicing these conversions until you become proficient.

Now that we've laid the groundwork for understanding the binary number system, let's talk a little about how the computer uses this information. Binary numbers are the bottom line when you're working with computers. Everything in the computer is built up from and eventually must be reduced to binary in order to be understood by the computer. But how the computer interprets the binary number depends on the context. For example, machine code, BASIC tokens, CBM ASCII, musical notes and screen characters all break down to binary numbers. But, depending on the context, a 1000 0111 can be either the f5 function key in CBM ASCII, the READ command in tokenized BASIC, the musical note "C" or, when POKEd to a screen location, a reversed "G". The context is established by the operating system of your computer.

Let's just take one example to show you how binary numbers are used to make the computer "think". The instruction set is all the commands that the 6502 is capable of executing. Each of these commands is a binary number. The 6502 microprocessor does not recognize decimal or hex numbers. For example, when the 6502 chip "sees" the binary number 10101001 it knows to "put the next number seen into the computer's accumulator". If the computer "sees" the 11011000 binary number it knows to "add one to the Y register".

To take this concept a step further—a program is a logical sequence of instruction codes (along with data and memory locations) that will make the computer perform a particular task. Here is a short program that will add two numbers together from register 30 and 31, then place the result into register 32.

Binary	Hex
1010 0101	A5
0001 1110	30
0110 0101	65
0001 1111	31
1000 0101	85
0110 0010	32

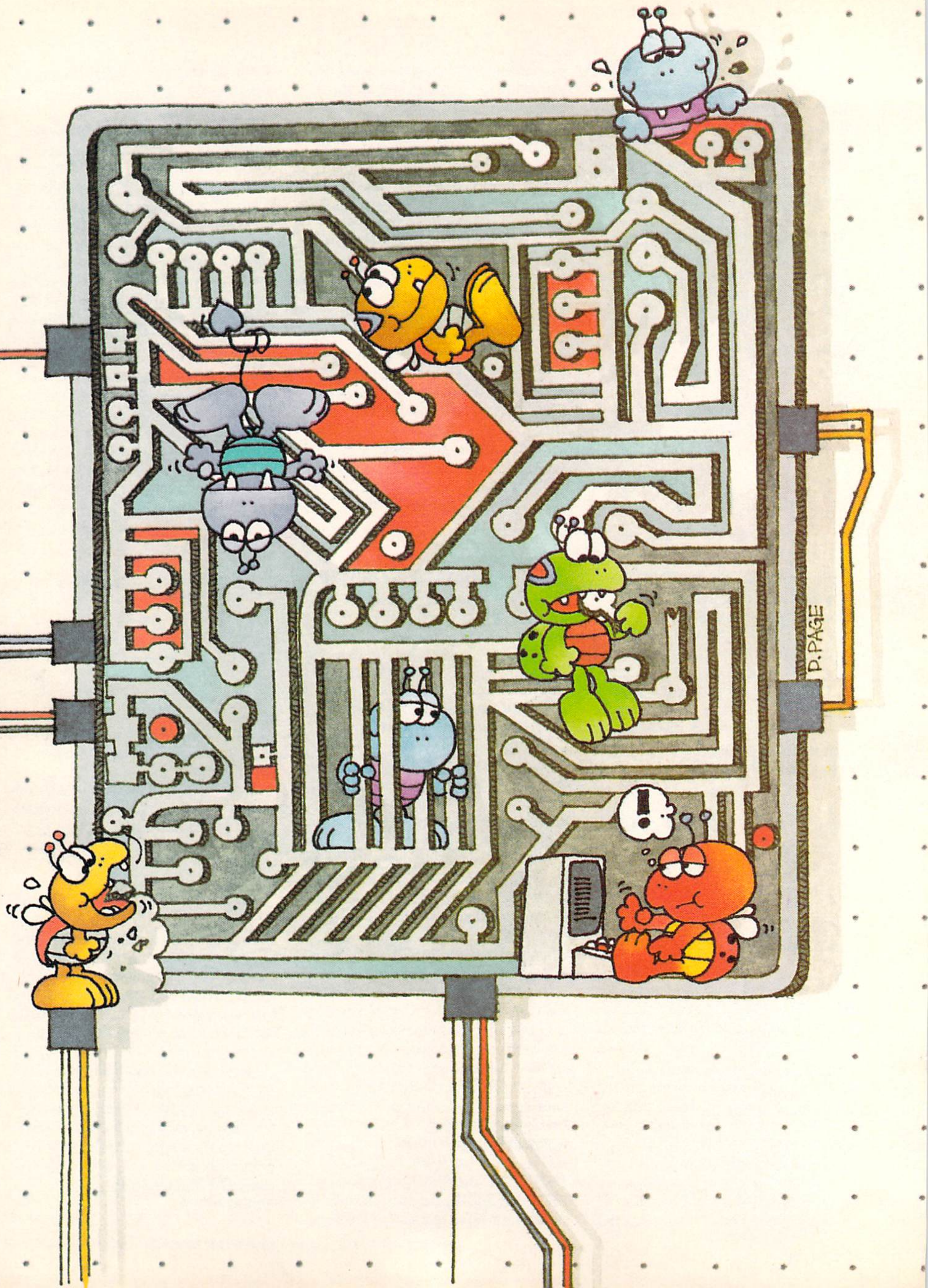
As you can see the binary code is difficult to read and the hex is slightly easier to read.

The next level above this is assembly language. This program in assembler would read as follows:

```
LDA     $30
ADC     $31
STA     $32
```

This is easier for people to understand, but the computer must still translate the code down to ones and zeros. And BASIC, which is at an even higher level than assembler, must also go through this translation process down to the binary level.

Hopefully, this will make computers more understandable to you, but we've just scratched the surface. In the next few articles, we'll touch on computer logic, some assembly language and eventually set up a model for 6502 computer architecture.
So stay tuned. **C**


!
P. PAGE

Exterminating 101
Or Debugging Programs Can Be Fun (Sort of...)

If you've never had a bug in a program, you can stop reading now. You're either a being from another world or you haven't ever touched a program, so it won't mean much to you anyway. However, for those mere mortals who have been involved in programming, even superficially, this article will be an immense help in finding—and exterminating—those "annoying, pesky, elusive" little creatures that inevitably foul up your best work.

BY JIM GRACELY

ILLUSTRATION—DON PAGE

Nothing in the world of programming can match the feeling of writing and entering a 150-line program..... and watching SYNTAX ERROR light up on the screen... over.... and over.... and over. These errors, which seem to evolve from the program, are affectionately referred to as BUGS (not short for anything). The name is quite appropriate. In fact they are everything that bugs should be: they're annoying, pesky, elusive, and whenever you completely wipe them out of one place, they mysteriously appear somewhere else. This article will examine bugs in an attempt to answer three questions about them: 1) where they come from, 2) how you find them and 3) how to exterminate them!

Bugs that appear in programs can be divided into three categories: syntax errors, procedural errors and analytical errors.

Syntax Errors

Syntax errors are probably the most common of all bugs. They are especially fond of magazine programmers (people who enter the long programs printed in magazines). Most of the time these errors are simply typing mistakes, but of course BASIC doesn't know that.

BASIC is programmed to recognize specific arrangements of characters as commands, and specific arrangements of commands as program lines. If a program line contains anything other than these specific arrangements, a syntax error will be generated when the line is executed.

Syntax errors are generally easy to find because BASIC keeps track of the line it is working on, and this line number is included with the error message. The most common syntax errors are a misspelled command, an "O" and zero switched, and a semi-colon in place of a colon.

There is only one syntax error that I know of that will not include the correct line number. If you make a syntax error in a DEF FN line, the syntax error will point to the line which contains the function call and not the DEF FN line.

Syntax errors can be prevented by carefully entering each line of a program. They can be exterminated once they appear by using the various editing capabilities of the Commodore computers.

Procedural Errors

Procedural errors occur when all the syntax of a program line is correct, but the line is attempting to do something that is not allowed. For example, if there is no line 20 in your program, the line GOTO 20 will be a procedural error.

After BASIC checks a program line for syntax errors, it tries to execute the line. Any errors that occur now are within one of the BASIC command procedures (GOTO, READ, IF..THEN etc...). BASIC keeps track of which procedure it is in, and uses that information to generate an error message. Like syntax errors, procedural errors will generate an error message including the line number that BASIC was working on. Unlike syntax errors, however, the reasons for a procedural error are not always as easy to detect. All the procedural errors and their causes are listed in the user guides for the VIC 20, Commodore 64 and PET/CBM computers.

The following list highlights some of the errors which may be harder to detect:

1) An "illegal quantity" error may be caused by POKing a memory location with a value of less than 0 or greater than 255.

2) An "out of data" error may be caused by attempting to READ a list of data a second time without first using the RESTORE command.

3) A "RETURN without GOSUB" error may be caused by letting the program run through a subroutine when the program should have ended. This will occur most often when subroutines are at the end of a program.

4) A "NEXT without FOR" error may be caused by switching variables in the NEXT statements (e.g., NEXT X instead of NEXT Y). BASIC will figure out which variable goes with each NEXT statement if there is no variable. This is an easy way to avoid this cause of an error.

Many procedural errors that may appear in a program can be found by examining the line referred to in the error message. Sometimes, however, a procedural error is simply the result of some other error. A "divide by zero" error may be easy to find, but its cause may be an error which falls into the next category.

Analytical Errors

Analytical errors are the cockroaches of programming bugs. These are errors that BASIC doesn't detect, and for this reason, no error messages are generated. Analytical errors are the hardest to find, the hardest to figure out and the hardest to exterminate. These errors may be caused by anything from an endless loop (10 X = 5 : GOTO 10) to a basic flaw in the equations the program is using (if you want the variable T to be the product of X and Y, the program line; T = X + Y will be an analytical error). There are three possible results of an analytical error:

1) A procedural error: As mentioned, a "divide by zero" error may be the result of not flagging a zero value. The following program is an example:

```
10 T=3
20 X=5
30 R=X/T
40 T=T-1:X=X-1
50 IFX<1THEN END
60 GOTO20
```

If you run this program you will get a "divide by zero" error in line 30. The error however, is not really a problem with line 30. The problem with the program is actually in line 60; the GOTO should be to line 30.

2) Incorrect results: This problem is often very frustrating. The program runs without any errors, but the result is wrong.

3) Nothing: To some people, this is the worst thing that can happen when they run their program. A program that never ends or ends without producing any results is the hardest program to debug. Without any results at all, it's hard to find a starting point for debugging.

In all three of these possible results, I've said nothing about how to find the bugs causing these errors. There are two methods that I will discuss to debug a program with analytical errors. I call the first method STOP-PRINT debugging. The second uses what is called a "trace" program.

STOP-PRINT Debugging

The idea behind this kind of debugging is to use STOP commands to halt program execution and then use PRINT statements to print any "active" variables. Active variables are those variables that change value as the program is executed. By checking the value of these variables at various places while the program is running, the location of a bug may be found. Here is one example of using this type of debugging:

```
10 INPUT"LENGTH IN FEET";A
20 INPUT"NUMBER OF FOLDS";B
30 R=1
40 A=A/2
50 IFR=BTHEN80
60 R=R+1
70 GOTO30
80 PRINT"NEW LENGTH
   IS "A" FEET"
```

This program asks for a length (of paper, cardboard, etc.) and a number of folds. It then computes the new length based on this information. At least that is what it is supposed to do. Enter the program and run it using 12' for the length and 4 as the number of folds. Nothing happens! Press RUN/STOP to halt the program. One good place to put STOP commands in a program is right before an IF-THEN statement. This is a place where a variable is checked, and before BASIC gets it, we want to look at it. So add "STOP" to the beginning of line 50. Now run the program using the same values. When the program breaks in line 50, type the following line.

```
PRINT R,A
```

It shows that R=1 and A equals 6 (half of 12). Type CONT (for continue) and when the program breaks again, enter the PRINT line again. This time R=1 and A equals 3 (half of 6). We could do the same thing again, but we've already found the problem. Line 60 adds 1 to the value of R each time A is halved. But the second time the program broke, R was still equal to 1. To check if line 60 is really working, put a ":STOP" after line 60, and remove the previous STOP added to line 50. Now run the program and when it breaks type PRINT R. Now R=2. But if R=2 now, and R=1 in line 50 then what is happening? It seems as if R is being reset to 1 each time through the program. This is exactly what is happening. Line 70 has a GOTO 30 that resets R! This line should be GOTO 40.

This is just one example of using the STOP-PRINT debugging technique. Debugging programs is like most skills in that the best way to learn is by doing. In fact, with this particular skill it is the only way. There are no secret formulas for debugging a program and the steps needed for each program are different than those for any other program. Listed below are some suggestions and hints for debugging programs using the STOP-PRINT technique.

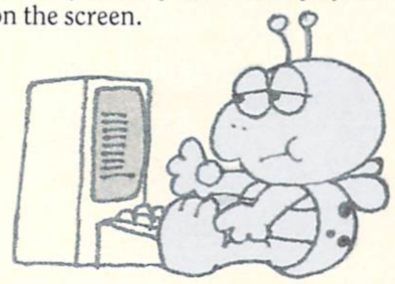
1) The STOP command can be used to determine where the flow of a program is *not* going. To do this, place a STOP command directly after each loop (FOR..NEXT or any others) within the program starting with the first loop to *end.* Continue moving the STOP command outward through all of the loops until you find a STOP command that isn't executed. This allows you to concentrate any further debugging on the loop that is not ending.
2) Don't use too many STOP commands at once. One or two is enough to keep the debugging quick, without getting too confusing. Two good places for STOP commands are before an IF..THEN statement and after a NEXT statement.

Trace Debugging

The second debugging technique, which I mentioned earlier, is to use a trace program. A trace program will "trace" or allow you to follow the program as it is being executed. The most advanced trace programs will display the changing values of every variable. The two programs at the end of this article are trace programs for the VIC 20 and the Commodore 64. Both programs are basically the same with changes to adapt them to either system. This program was originally written by Nick Hampshire to operate on the PET, and first appeared in his book *The PET Revealed.* These revised versions appear with the kind permission of Nick Hampshire.

This trace program uses the CHRGET routine in page zero memory to print each character of each line of the program being traced to the screen *before* BASIC gets it. The trace program creates a two- (Commodore 64) or three- (VIC 20) line field at the top of the screen, which displays the current line.

The program is quite nice for debugging large programs when you have no idea where to start. One of the interesting features of the program is that it displays the value or character(s) read by a READ statement. You can also see when an IF..THEN statement is false, because if the IF portion is false, BASIC won't even look at the THEN portion, and only the IF portion is displayed on the screen.

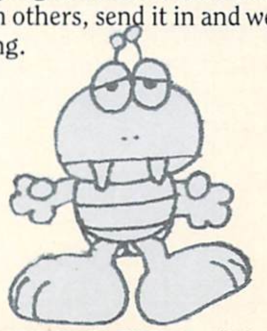


Here are some instructions for using the trace program. Enter the program for your computer and save it. Now type RUN. After a couple of seconds the screen will display three SYS commands and a POKE location that will control the trace program. Write down the locations and what they are used for. The program resets all the BASIC pointers and POKEs the machine language program into memory. On the Commodore 64, the program resides between 52658 and 53002. This preserves all 38911 bytes of BASIC ROM for the program to be traced. On the VIC 20, the program resides between 7337 and 7679. This leaves about 3200 bytes of BASIC ROM for the program to be traced.

Type NEW to erase the BASIC portion of the trace program. Use the initialize SYS command and then type NEW again. This initializes the trace program, but still allows normal operation of the computer. Now load or enter the program you want to trace. Use the enable SYS command to turn on the trace program, and to give it control of the CHRGET routine. Now when you type RUN, the trace program will display the line BASIC is working on. The program being traced will run normally, although at a much slower rate. RUN/STOP also still functions so that program execution can be halted at any time. After running the trace on a program, use the disable SYS command before editing the program or loading a new program. When you want to trace again just use the enable SYS command.

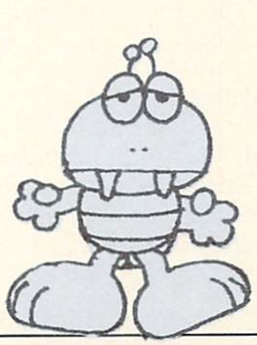
The speed of the trace program can be controlled by POKing various numbers to the POKE location that was displayed. The default value in this location is a five. Decreasing this value will speed up execution and increasing the number will slow down execution.

Although debugging programs usually isn't a lot of fun, it is always a way to learn. Understanding why errors occur helps to eliminate them from your programs, and the less time you spend debugging one program, the more time there is to work on another! The debugging techniques presented in this article only reflect my personal preferences and are by no means the only ways to debug programs. If you have your own special technique or trace program and would like to share it with others, send it in and we'll pass it along. **C**



```
1 REM ***TRACE FOR VIC-20***
2 REM ***WRITTEN BY NICK HAMPSHIRE***
3 REM ***MODIFIED BY JIM GRACELY***
10 E=55
15 D=2
100 DATA-342,162,5,189,135,227,149,115,202,16,248
110 DATA169,239,133,131,96,173,-342,133,55,173,-341
120 DATA133,56,169,255,133,45,160,0,162,3,134,46
130 DATA162,3,32,-271,208,249,202,208,248,32,-271
140 DATA32,-271,76,142,198,162,5,189,-6,149,115,202
150 DATA16,248,169,242,133,131,96,230,45,208,2,230
160 DATA46,177,45,96,230,122,208,2,230,123,96,32
170 DATA118,0,8,72,133,215,138,72,152,72,166,58,165
180 DATA57,197,251,208,4,228,252,240,106,133,251
190 DATA133,38,134,252,134,39,169,5,133,110,202,208,253
200 DATA136,208,250,234,234,198,110,208,244
210 DATA234,160,44,169,160,153,255,29,138,153,255,149
220 DATA136,208,244,132,254,132,40,132,41,132,42
230 DATA120,248,160,15,6,38,38,39,162,253,181,43
240 DATA117,43,149,43,232,48,247,136,16,238,216
250 DATA88,162,2,169,48,133,106,134,105,181,40,72
260 DATA74,74,74,74,32,-44,104,41,15,32,-44,166
270 DATA105,202,16,233,32,-38,32,-38,165,253,197
280 DATA122,240,55,165,215,208,4,133,251,240,47
290 DATA16,42,201,255,208,8,169,105,32,-30,24,144
300 DATA33,41,127,170,160,0,185,157,192,48,3,200
310 DATA208,248,200,202,16,244,185,157,192,48,6
320 DATA32,-32,200,208,245,41,127,32,-32,165,122
330 DATA133,253,104,168,104,170,104,40,96,168,173
340 DATA4,144,41,127,208,249,152,96,9,48,197,106
350 DATA208,4,169,32,208,2,198,106,41,63,9,128
360 DATA132,109,32,-54,164,254,153,0,30,192,195
370 DATA208,2,160,7,200,132,254,164,109,96,76
380 DATA-255,32,-262
400 S2=PEEK(E)+256*PEEK(E+1):S1=S2+D-344
410 FORJ=S1TOS2-1
420 READX:IFX>0ORX=0THEN450
430 Y=X+S2:X=INT(Y/256):Z=Y-X*256
440 POKEJ,Z:J=J+1
450 POKEJ,X
460 NEXTJ
500 PRINT"INITIALIZE WITH SYS(";S1+17")"
510 PRINT"ENABLE WITH SYS(";S1+56")"
520 PRINT"DISABLE WITH SYS(";S1+2")"
530 PRINT"CHANGE SPEED WITH POKE";S1+121-D",X"
600 END
```

```
1 REM ***TRACE FOR C64***
2 REM ***WRITTEN BY NICK HAMPSHIRE***
3 REM ***MODIFIED BY JIM GRACELY***
15 D=2
100 DATA-342,162,5,189,162,227,149,115,202,16,248
110 DATA169,239,133,131,96,173,-342,133,55,173,-341
120 DATA133,56,169,255,133,45,160,0,162,3,134,46
130 DATA162,3,32,-271,208,249,202,208,248,32,-271
140 DATA32,-271,76,142,166,162,5,189,-6,149,115,202
150 DATA16,248,169,242,133,131,96,230,45,208,2,230
160 DATA46,177,45,96,230,122,208,2,230,123,96,32
170 DATA118,0,8,72,133,215,138,72,152,72,166,58,165
180 DATA57,197,251,208,4,228,252,240,106,133,251
190 DATA133,38,134,252,134,39,169,5,133,110,202,208,253
200 DATA136,208,250,234,234,198,110,208,244
210 DATA232,160,80,169,160,153,255,3,138,153,255,215
220 DATA136,208,244,132,254,132,40,132,41,132,42
230 DATA120,248,160,15,6,38,38,39,162,253,181,43
240 DATA117,43,149,43,232,48,247,136,16,238,216
250 DATA88,162,2,169,48,133,106,134,105,181,40,72
260 DATA74,74,74,74,32,-44,104,41,15,32,-44,166
270 DATA105,202,16,233,32,-38,32,-38,165,253,197
280 DATA122,240,55,165,215,208,4,133,251,240,47
290 DATA16,42,201,255,208,8,169,105,32,-30,24,144
300 DATA33,41,127,170,160,0,185,157,160,48,3,200
310 DATA208,248,200,202,16,244,185,157,160,48,6
320 DATA32,-32,200,208,245,41,127,32,-32,165,122
330 DATA133,253,104,168,104,170,104,40,96,168,173
340 DATA18,208,41,255,208,249,152,96,9,48,197,106
350 DATA208,4,169,32,208,2,198,106,41,63,9,128
360 DATA132,109,32,-54,164,254,153,0,4,192,195
370 DATA208,2,160,7,200,132,254,164,109,96,76
380 DATA-255,32,-262
400 S2=53000:S1=S2+D-344
410 FORJ=S1TOS2-1
420 READX:IFX>0ORX=0THEN450
430 Y=X+S2:X=INT(Y/256):Z=Y-X*256
440 POKEJ,Z:J=J+1
450 POKEJ,X
460 NEXTJ
500 PRINT"INITIALIZE WITH SYS(";S1+17")"
510 PRINT"ENABLE WITH SYS(";S1+56")"
520 PRINT"DISABLE WITH SYS(";S1+2")"
530 PRINT"CHANGE SPEED WITH POKE";S1+121-D",X"
600 END
```



The CHEM-PUTER Laboratory

by Curtis T. Sears, Jr.
Department of Chemistry, Georgia State University

Commodore PETs help teach college chemistry—and may lower the failure rate for freshman chemistry courses.

Georgia State University, located in downtown Atlanta, is an urban commuting school and the second largest institution of higher learning in the state. The average age of the 22,000 member student body is 28, somewhat above the national average for college students. Classes begin at 7 am and continue until 10 pm in order to accommodate the needs of the students, most of whom hold full or part-time jobs.

In conjunction with some committee duties in 1978, Fred Henneike, a fellow chemistry department faculty member, and I were discussing a problem common to college and university chemistry departments throughout the country—the very high failure rate in the first term freshman chemistry course. Numerous nationwide studies had documented that approximately 35 percent of the students enrolling in the first term of general chemistry do not achieve a C grade or better on the first attempt. This is about twice the average rate for all freshman college courses. Our department's rate was hovering at 45 percent, fully 10 percent above the national average. Our discussion centered on possible reasons for the difficulty experienced by our students and ways we might improve our students' chances of success while maintaining the course standards.

It appeared to us that a combination of factors was contributing to the problem. The most important were: (1) increased class size; (2) little out-of-class contact between students and faculty; (3) lack of an adequate mathematics and science background.

Over the decade from the late 1960's to the late 1970's, class sizes nearly tripled from an average of approximately 30 to about 80 students. The increased class size was accompanied by a change in emphasis from classroom discussion to highly structured lectures and the cessation of mandatory homework assignments.

In 1970, over 80 percent of the students enrolling in general chemistry had taken a high school chemistry course. But because of the liberalization of high school graduation requirements, by 1978 under 40 percent of our students had had any previous exposure to chemistry. The adequacy of their preparation for college chemistry was weakened further by an eight-year average time lag between high school graduation and enrollment in the course. Further aggravating the situation, opportunity for out-of-class aid from the faculty is generally more limited at commuter schools than at residential ones and is further restricted at GSU by work and family demands on our older student population.

Chemistry students at Georgia State University use Commodore PETs as tutors.

Once we identified these impediments to student progress, we decided that any solution must economically provide: an approximation of the individual attention students previously received in smaller classes; a suitable replacement for the immediate feedback given in the past by the return of graded homework; material designed to help the poorly prepared overcome their deficiencies; and availability at times convenient for the student. We thought a properly designed and implemented computer-assisted instructional program should be able to meet these objectives.

Both of us had made several previous excursions into computer-assisted instruction (CAI) on mainframe computers. We had consistently been forced to retreat in the face of an insufficient number of terminals to handle 800-1000 students per week, the lack of lower case letters or sub or superscripts, and the computer's inability to reject unacceptable student input. The combination of these things had served to frustrate rather than educate our students.

By 1978, however, we felt the newly available microcomputers could truly make CAI feasible. Accordingly, we prepared and submitted a proposal to the National Science Foundation for support of microcomputer-aided chemistry instruction. Revision of our initial proposal eventually led in June 1981 to a grant from the Comprehensive Aid to Undergraduate Science Education program of NSF in partial support of a $315,000 project in chemical CAI to be developed over a 42-month period.

The first task confronting us after we were notified of the award was the selection of the microcomputer to be used. After two months of talking with people throughout the country and testing a large number of machines, we selected the Commodore PET model 4032. We chose the PET because it has a one-piece unit that makes transport and security easier, a built-in upper/lower case capability and character graphics, the ability to easily display sub and superscripts through the imbedding of cursor movements in data statements, a separate numeric keypad, and is generally sturdy. Additional enticements were the 3-for-2 offer then in effect from Commodore and the availability of a hardware-controlled network to allow the PETs to share disk drives and printers.

During the fall of 1981 the university renovated a little used faculty and staff lounge in a major classroom building to create a pleasing environment specifically designed for our CHEM-PUTER laboratory. In 1982 the worst winter storm to hit Atlanta in over 30 years arrived on the same January day as our thirty PETs, ten 8050 dual disk drives, five printers, seemingly unending coils of cables, stacks of manuals, boxes of assorted bits and pieces, and an engineer from NEECO whose assigned task was to create a functioning microcomputer installation and within 24 hours make us experts in its operation.

With the outside temperature in the low teens we quickly discovered why the lounge had been little used. The room, located on the north side of the building with floor-to-ceiling plate glass windows, had no heat. Dressed in heavy winter jackets and lined gloves, the three of us unpacked and set up equipment. Miraculously, by late afternoon we had a tested and properly functioning microcomputer laboratory. Throughout the day as Fred and I muttered about the cold, our man from NEECO good-naturedly chided us about being transplanted northerners softened by a decade in Atlanta. His patience, cheerfulness and helpfulness we have since recognized is typical of all the personnel at NEECO. They have unfailingly answered all our many questions, made helpful suggestions and serviced our equipment promptly and courteously. I have dealt with many vendors of many products and these folks are tops. I wish I could convince the service manager at my automobile dealer to spend a week with Bob Crowell!

Where are we now, a little over a year later and nearly halfway through the project? Our room now has heating and air-conditioning, and we are serving about 25% (200-250 individuals) of our beginning chemistry students. Our program library contains in excess of 300 programs including instructional programs, utilities, programmer's aids, record-keeping and games. In addition to our own programs, we have acquired programs from other institutions, commercial sources and user groups.

As a result of our experience I urge anyone using microcomputers in education to join user groups. Two from which we have benefited greatly are the Central Illinois PET Users Group and the Toronto PET Users Group. Both of these publish newsletters and maintain extensive libraries of public domain programs that can be obtained very inexpensively. They also publish no-nonsense reviews of hardware and software. These reviews alone can easily save you several times the membership cost.

Currently student use of the programs is, with one exception, voluntary. We simply inform the students of the existence of the facility and give them a brief description of the programs available. The reasons for this are the limited number of programs available and, until this month, a lack of record-keeping abilities. Very few of the programs obtained from outside sources meet our criteria for completeness of topic coverage, ease of use and educational value. Consequently, we decided not to force their use on our students. Because our students' average age is higher than most, with an accompanying anxiety toward computer use and increased scheduling difficulties, we believe no experience is better than a poor first experience.

However, we are designing many programs ourselves. There are nine lecture and three laboratory courses served by the microcomputers. We have authored fifteen programs for use in three courses. The faculty regards this as an insufficient number to incorporate into the courses as routine required assignments at this time. But, since our rate of program production has now reached one completed program every three weeks, by the start of the new academic year in September we will satisfy faculty requirements for the number of programs available and they will become required assignments at that time.

The other factor delaying mandatory program use has been record-keeping. Without a means of quickly and easily identifying the student, the program used and the progress made, it is impossible to keep track of required assignments. We have just installed a record-keeping program and will be testing it throughout the spring and summer terms. When class registration is completed, we access the enrollment information maintained on the mainframe computer, and create a file for each student. The information contained in each file consists of student name, student social security number, the course number, the instructor's name and 40 fields that will eventually be filled with information on student program use.

A student wishing to use a program will be prompted to enter the course number, professor's name and his/her social security number. In progression we locate the disk files for the course, select those for the professor named and finally search this group for the particular student. If a match is found, the student's name is displayed on the screen and the student is queried if it is correct. The prior breakdown by course and professor minimizes the search time. If the student name is incorrect or no match is found, the student is instructed to see the room attendant who will either add the student to the files or determine the entry error. When a match is attained, a menu of available programs relevant to the particular course is displayed. After the program choice is made via a single keystroke, it is automatically loaded and run. When the program is exited, a disk record is made of the program used, time spent, the number of problems attempted and the number correct. An alphabetical listing can be prepared weekly for the professor detailing the record of each student in the class.

In addition to the two of us we have four part-time programmers. They are junior and senior information science students, none of whom has any chemistry background. Working twenty hours per week, each completes a debugged and tested program every eight to ten weeks. The coding for a typical program occupies 25-28K of memory and requires 2-5K of variable space to run. It has been necessary to compact several programs and reset BASIC pointers to effectively eliminate DATA statements after arrays have been filled in order to provide sufficient memory for the program to run.

Each program covers a single chemical concept in depth. Because of the existence of dual- and triple-track courses, a particular concept is often a part of two or three courses. Programs on such concepts are written to contain levels of concept development appropriate for each of the courses. A program menu displays the levels available, and the student chooses the level appropriate to the course in which he/she is enrolled. Most programs have three parts: (1) concept review and examples of its application; (2) a tutorial section in which the student is guided through a problem to its solution; (3) drill and practice. A student may move to any of the parts at any time. If a student's performance falls below a certain percentage of correct responses, the student is forced into the tutorial section. A biaser in the random problem selector presents increasingly more difficult problems when the student's performance exceeds a specified percentage of correct responses.

Intelligent input routines monitor the attempted student input and reject inappropriate input such as no input, letters or characters for numbers and vice-versa, thereby preventing most program crashes. The advantage a computer offers over a problem book with answer keys is the ability to analyze an incorrect response. All student responses are checked for the most common types of error. For example, two numbers may have been multiplied when one should have been divided by the other. When such an error is detected, an error message is given with a brief explanation of why the response is incorrect.

Program development begins with a faculty committee. This committee selects the topics, writes a summary of what each program is to do and assigns a priority to each. I take over the program at this point, totally designing its functions and presentation. This includes the data base, the concept review, examples, tutorials, drill problems, and response to various student inputs. We have printed grid sheets representing the screen, and the location of every character to be displayed is specified. Thus, before any coding is done, we know exactly what the presentation will look like to the user.

Average program specifications are 50-60 screen display sheets, plus another 20-30 pages of instructions and data. I have a fair knowledge of the capabilities of the computer and its operation, but scant knowledge of programming details. This turns out to be a great advantage because it frees me to focus exclusively on the educational value, appearance, and ease of use. I remain almost totally free of concern about programming ease or expediency. When a programmer comes to me with the comment, "if you would change such-and-such feature of the program, it would make the programming faster... easier... or...," my normal response is "what you mean is that you're feeling lazy today" or "you're simply not being clever enough to accomplish the task." Needless to say, the programmers seldom bring such things to my attention any longer.

When my work is done, Fred takes over. He goes through the design, constructs program flow diagrams, makes suggestions for programming approaches to particularly knotty problems and decides if certain routines need to be accomplished via machine language. He schedules an appointment with the programmer to go over the program. Any machine language portions, he retains and programs himself. The programmer is later given a copy of the routine and instructions for its incorporation into the program.

The finished product is distributed to the faculty committee for their review and suggestions. Suggested revisions are discussed and any necessary modifications made. The program is then made available for student use. Each student using the program completes an evaluation questionnaire and selected students are interviewed by a separate evaluation committee. I am delighted to report that the students generally evaluate our programs very highly and feel they have made a substantial contribution to their learning. The main complaint we receive is, "why isn't there a program for_______________ yet?"

It has been an exciting nineteen months for us, and we anticipate the remaining time on this project will be both rewarding and challenging. Persons interested in visiting our CHEM-PUTER laboratory or obtaining further information may write the author at the Department of Chemistry, Georgia State University, Atlanta, Georgia 30303.

This material is based upon work supported by the National Science Foundation under Grant No. SER-8103942. Any opinions, findings and conclusions or recommendations expressed in the publication are those of the author and do not necessarily reflect the views of the National Science Foundation. C

Computer Case Company

AN OPEN AND SHUT CASE FOR COMPUTERS

CM703

Your sizable investment in your computer should be protected. If you have to load and unload your computer every time you want to move it, you must consider the possible damage. Computer Case Company has solved your computer protection problem. Once your equipment is safely inside the attache-style carrying case, it never has to be taken out again. Simply remove the lid, connect the power, and operate. For storage, disconnect the power, enclose your disks, working papers, and manuals in the compartments provided, and attach the lid. Case closed. Judge for yourself.

● CM703	Commodore 64/VIC with Disk Drives	$129
● CM704	Commodore 64/VIC with Dataset	109
● AP105	12 inch Monitor (B&W)	99
● AP106	AMDEK Color I, II or III	119
● P401	Paper Tiger 440/445/460	99
● P402	Centronics 730 Series	89
● P403	Epson MX70, MX80	89
● P404	Epson MX100	99
● P405	IDS 560 or Prism 132 Printer	109
● P406	Starwriter F10 Printer	119
● CC80	Matching Attaché Case (5")	85
● CC90	Matching Attaché Case (3")	75
● CC91	Matching Accessory Case	95
● CC92	5.25 Diskette Case	49

CALL TOLL FREE: (800) 848-7548

5650 Indian Mound Court • Columbus, Ohio 43213 • (614) 868-9464

WE HAVE WHAT YOU NEED . . .

COMPUTER MARKETING

the

foxiest

PRODUCTS FOR THE COMMODORE 64!™

Call Us Today For Your Nearest Commodore Dealer...
You'll Be So Glad You Did!

Distributed By:
COMPUTER MARKETING SERVICES INC.

300 W. Marlton Pike
Cherry Hill, New Jersey 08002
(609) 795-9480

Commodore 64 is a trademark of Commodore Electronics Limited

Six Months with a 64

by Doris Dickenson

In early 1982, fourth grade teacher Doris Dickenson wrote an essay titled "Why I Want to Go Back to School with a Microcomputer" for a contest run by Instructor Magazine*—and won herself a Commodore 64 computer. So in the fall, she actually did go back to school with that micro—the only computer in her California school district.*

With no experience and little access to other computer users, Doris did what pioneers have always done; she quickly devised her own methods, using whatever she had on hand. One of the products of her resourcefulness was a children's user guide for the 64, You and Your Computer. *(See page 71 for Part 5 of this guide.)*

In this article Doris offers advice to teachers who may want to teach computing, but are inhibited by lack of experience and/or a limited budget. We all hear about the school boards that allot big budgets for advanced computer curricula, but the fact is that many school districts are not yet able to go that route. If your district, like Doris', is one of those that hasn't had the money to devote to extensive computer courses, you don't have to be left out of the computer revolution. Doris can vouch for that.

There is so much going on when you have a computer in the classroom I find it hard to believe it has only been six months since I "went back to school with a microcomputer", thanks to an essay contest run by *Instructor Magazine*. Having only a little personal experience with computers, and with no other computers in the district, I had very few preconceived ideas about what to do with my own Commodore 64 once I won it.

Common sense told me not to just hand the computer over to my fourth grade students without some preliminary explanation and standards. I wanted it to survive their first burst of enthusiasm. I guess I imparted my concerns to them because they have always been very responsible in its use. The one episode that disturbed me frayed my nerves more than it damaged the computer.

Not having any separate memory device at first, we could not save any programs or games, so my first step was to help the students become familiar with the computer itself. Using some recommended computer literacy materials, I did a few lessons with the entire class on computer hardware and vocabulary as an introduction. Then, using the manual that accompanied my 64, but which was too difficult to present to 9 and 10 year olds, I wrote several booklets, each dealing with one aspect of the computer itself, giving careful explanations and extending activities with further practice and open-ended student activities. My objective was to give them the chance to work independently in our Computer Corner.

One reason for this approach was that, realistically, you can't give 28 students enough individual teacher-time to get them "off the ground" with the computer—without sacrificing the rest of the curriculum and the other students. These circumstances strengthened my philosophy of classroom computer education—learning to use the computer as a tool for your own tasks, rather than as a "drill-and-practice" machine. A fellow teacher calls me a "purist".

Actually, this approach is one of the sources of disappointment for me. I have been both surprised and disappointed in student interest in the computer. I have found that my students commonly regard a computer as a game machine and prefer to interact with it, rather than to learn to have it work for them through their own input (programming). There seems to be a wide range of feeling about computers. Some students are merely curious, some remain timid, even after much practice, and some want more than their share of turns with it. I am not always fair about sharing turns. I tend to permit those who want to do their own programming to use it as much as possible.

The addition of a datassette, and

later a disk drive, made some prepared programs available to us. For speed and ease of access, I would not want to be without my disk drive, but for peace of mind, with 9 and 10 year olds using the equipment by themselves, I find a datassette invaluable. I can select the programs I want from disks, modify them if I choose, and then make them available for students at the Computer Corner. By posting simple, step-by-step directions, students can use the datassette on their own. A word of warning! Be sure to number each tape and number a list of programs so you don't LOSE programs. Cassettes can really get confused.

Now that software is becoming available for the 64, I still put low priority on drill-and-practice programs. However, I have found several public domain programs that enable me to plug in specific skills for my own curriculum, and I prefer these to a completely prepackaged program. I have also found that students can work with programs at levels higher than they would normally work, with the aid of "prompts", such as maps, lists, or information cards that can be used with some of the programs. It helps them score high, or beat the computer, which they love to do, as well as learn new concepts.

After using my Commodore 64 for the past six months, I have found some methods that have been helpful to me and might be helpful to others. I think that those of us who are starting to explore the use of classroom computers need all the help we can give each other. We are finding that out in our own school, where four more teachers have just added computers to their classrooms.

1. For a teacher to get acquainted with a computer and really learn to enjoy it, requires lots of "at home" time. Most teachers will find that they want to use computers during evenings or weekends. If they don't have one of their own, they will probably consider using the one from their classroom. My computer commutes with me almost daily. I saved the styrofoam packing case that it came in and find that it makes an excellent, dustfree carrying case. I cut two thin strips of truck inner tube and use them like rubber bands around the case.

2. If you have a disk drive, it comes in the same kind of styrofoam case and can also be carried around safely with the same kind of inner tube bands as above.

3. If you move your computer between different T.V.'s, used as monitors, a small sewing machine screwdriver is easy to carry around for attaching the T.V. switch box to the different sets.

4. Make a real Computer Center out of your computer. Put it near a bulletin board that will not only identify the area, but will give you an opportunity to mount a variety of computer-related displays.

5. To avoid the numerous cords and extensions that often are needed as you add peripherals to your classroom installation, a four-outlet power strip is a practical and inexpensive item. Some even have fuses in them.

6. How about keeping a small box at your computer center to hold a collection of computer related reading or browsing books for students to use while they are waiting for their turn on the computer. Inexpensive computer dictionaries or storybooks can be purchased, but you and your students can make your own booklets of computer pictures, computer stories and drawings, or examples of computer uses.

7. You will frequently be hearing the question, "Shall I turn the computer off?" and sooner or later someone will do just that, losing a program you had hoped to save. I solved the problem by folding a piece of heavy tagboard, about 8"×24", into a permanent sign that I leave standing at the Computer Corner. Make a crease across the tagboard at 4", 12", and 20". Overlap the two 4" ends and staple closed. This makes a wedge-shaped sign. On one side, in LARGE letters, print "Do NOT turn the computer OFF". On the reverse side, print "Do not turn the computer on". I have found that there are times when this is the appropriate message for my students, also.

8. If your students are working at the computer independently, you will find yourself wondering how they are doing, while you attend to other tasks in the classroom. It is a simple matter to attach a red HELP flag to a new pencil and make a small hole in the wedge-shaped sign. Then, if they need to have some help and want your attention without interrupting you, they put the sign on top of the monitor. You can be aware of a problem and go to them as soon as you are able.

9. When you hand-copy programs for your students to type into the computer, as you will do when you find them in books or magazines, or when you exchange programs with other teachers, use graph paper, 1/4" grid. This really simplifies the correct placement of spaces, which often can be critical.

More and more, as this school year progresses, I find that I am reviewing and reorganizing my computer education activities of the past six months. So many of them have been trial and error. I am looking forward to putting this year's experiences into practice next year, and to all the new things that are in the future for my 64—especially increased availability of software, and the introduction of LOGO. With computers, it seems as if you never stop learning, and things never stop happening. **C**

Don't just save money at K-12 MicroMedia's Software Sale

Because of our recent high volume purchase of *Creative Computing's* **– *Learn Something.***
best educational and game software, we can offer you these well reviewed programs *well below list price* for *free no-risk 30-day evaluation.* So if you've been waiting for a chance to try before you buy, you've come to the right ad. And, because we have better things to do with our time than process a pile of returns, we're pretty sure that you're going to like what you see. Let us know.

SAVE 45%

PET Programs on Cassette

Board Games (8K)	$7.00
Conversational Games (8K)	7.00
Ecology Simulations I (16K)	12.50
Ecology Simulations II (16K)	12.50
Graphic Games (8K)	7.00
Haunted House Adventure (16K)	7.00
Sector 3 Air Traffic Controller (24K)	7.00
Social & Economic Simulations (16K)	12.50
Study Made Easy (16K)	12.50
Trucker/Streets of the City (32K)	12.50

Official school purchase orders and preview requests on school letterhead are welcomed. Individual orders must be prepaid. Include 5% for shipping and handling (minimum $2.00 charge). New Jersey residents include applicable 6% sales tax.

K-12 MICROMEDIA INC

172 Broadway, Woodcliff Lake, N.J. 07675 (201) 391-7555

Do you own a commodore COMPUTER ?

We handle everything for the Commodore™ computer: Hardware, Software, Games and Books.

Everything for: PET,® CBM,™ VIC 20™

List your interests and equipment and mail to:

M.I.K. INC. USA
P.O. Box 4383
N. Hollywood
California 91607 USA
(213) 982-3403

PET is a registered trademark of Commodore Business Machines, Inc.
CBM and VIC 20 are trademarks of Commodore Electronics Ltd.

What does **COMMODORE** have that Apple, Radio Shack and IBM don't?

backpack™

BATTERY BACKUP SYSTEM

For CBM™/PET® 2000, 4000, 8000, and 9000 series computers and CBM 4040/8050 dual disk drives. Installs within the cabinets of the computer and disk drive. Recharges continually from the machine's own power supply and automatically supplies 30 minutes (max.) of reserve power during outages. Also eliminates surges and spikes. User installable.

OFFICIALLY APPROVED BY COMMODORE

In Canada call: Van-Hoy Group (604) 542-1138 or (604) 545-0794
In United Kingdom call: Wego Computers (0883) 49235

ETCETERA OF CSC CORPORATION
APEX, NORTH CAROLINA, U.S.A.
(919) 362-4200

SOLD ONLY BY INTELLIGENT COMPUTER DEALERS • DEALER INQUIRIES WELCOME

Multi-purpose CBMs Keep Student Interest High

by Jean Spahr
Librarian, Red Bud High School

Students don't want to leave their computer classes at this Illinois high school, where CBMs are used for everything from teaching office skills to running agricultural programs.

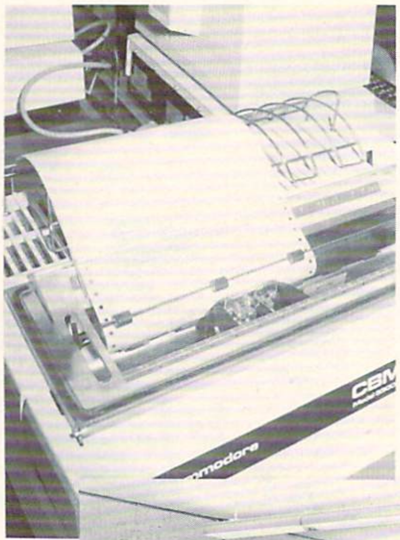
Red Bud High School is located in Red Bud, Illinois, a town of about 2,900 people approximately forty miles southeast of St. Louis, Missouri. The high school has an enrollment of about 500 students. RBHS is well-known locally for its excellent vocational and academic programs. The school has placed first in the nation for the past two years in the Test of Engineering Aptitude, Math and Science (T.E.A.M.S.) sponsored by the Junior Engineering Technological Society (J.E.T.S.).

In the spring of 1982, the Board of Education needed to replace manual typewriters with electrics in one of the business classrooms as well as to purchase microcomputers for a new programming course. Teachers in the business and agriculture classes had also expressed interest in using computers for instruction in their classes. Since the computer keyboard is remarkably like the keyboard of a typewriter, the Board felt it worthwhile to explore the possibility of using computers to teach advanced typewriting. The same equipment could then also be used for programming and agriculture classes as well as data processing and accounting classes. If this were possible, the purchase would be very cost-effective because one room of equipment could serve many purposes.

After months of phone calls to educators, consultants, and computer dealers, the district decided that what it had in mind could be done. In the summer of 1982, a computer laboratory was installed. The Commodore 8032 was the model chosen because it has an eighty character line and upper and lower case capabilities, which were needed for the typewriting class. To avoid purchasing disk drives for each machine, two MUPET II systems were used. Each MUPET links ten computers to a dual disk drive and a printer. The Word Pro 4+ word processing program is used with the system in the typing class.

The equipment is now fully operational. Reports from teachers who are using the lab are very favorable. Student interest is high. In one of the classes the teacher reports that students do not want to leave when the class period is over. Students feel that using the computers has given them a realistic picture of the modern office. Although there have been some minor problems with the MUPET system, there have not been any repairs on any of the Commodore equipment.

The transition into the computer age has been relatively smooth. The success of the program at Red Bud High School can be directly attributed to a progressive and supportive board of education, an enthusiastic administration, and consultants and dealers who are willing to share their information. Most of all, the success can be attributed to interested and cooperative teachers who want to prepare their students for the age in which they will live and work—the age of computers. C



You and Your Computer
Part 5—Color Me Purple

by Doris Dickenson

As it turns out our inimitable fourth-grade teacher in Sebastopol, California, has more chapters to her children's user guide for the Commodore 64 (last issue we mistakenly announced the conclusion of the series). Here in Part 5 young students learn to use both the color control keys and the POKE command to program in color.

Your Commodore 64 can make lots of color without much effort. Here are some simple programs to give you ideas to start with.

A. Writing With Color

Do you remember how you used the CTRL and "Commodore" keys with the number/color keys in "You and Your Computer," Part 1? You can use these same keys inside a PRINT statement to write words in color. The color chart is on page 57 in the *Commodore 64 User's Guide.*

NOTE: Don't worry if your typing looks funny on the screen. Some unusual symbols will be printed for the colors, but your program will automatically change the colors.

1. Type this program using your own name and other information in the blanks.

Use the CTRL and C= keys

```
NEW
10 PRINT "CTRL 5 MY NAME IS ________."
20 PRINT " C= 6 MY SCHOOL IS__________
______________________________________."
30 PRINT "CTRL 8 I AM ______ YEARS OLD."
40 PRINT " C= 1 I LOVE TO EAT_________
______________________________________."
50 PRINT "CTRL 3 THIS LOOKS LIKE
  A RAINBOW."
60 END
RUN
```

2. Questions:
   a. You will see that some colors are easier to read than others. What do you think will happen if you change one of the colors to CTRL 7? Try this to find out.
   b. Press RUN/STOP and RESTORE together. Type LIST and make this change.

Use key

```
Type: 30 PRINT "CTRL 7 I AM ______
           YEARS OLD."
      RUN
```

What happened? Why did it happen?
Hint: What color does CTRL 7 print?

B. Color Bars

You can also type a program that will draw color bars that look like the ones in Part 1 by using a special computer code, CHR$, followed by a number in parentheses ( ). These are the special CHR$ (character string) numbers for the colors you can use:

(5) white	(28) red	(30) green
(31) blue	(144) black	(156) purple
(158) yellow	(159) cyan (blue)	

1. Type this program to print the color bars automatically:

```
NEW
10 REM: AUTOMATIC COLOR BARS
20 PRINT CHR$(147)
30 PRINT CHR$(18)"";
40 PRINT CHR$(5)"          ";
```

REM means REMARK. This reminds you what the program is and will not print.

This is another way of writing CLR HOME into your program.

This will print bars.

The space between the quotation marks sets the length of each bar. Try anywhere from 1 space to 40 spaces.

```
50 PRINT CHR$(28)"      ";
60 PRINT CHR$(30)"                  ";
70 PRINT CHR$(144)"         ";
80 PRINT CHR$(31)"          ";
90 PRINT CHR$(156)"           ";
100 PRINT CHR$(158)"        ";
110 PRINT CHR$(159)"            ";
120 END
RUN
```

2. Wasn't that easy? But why not fill the *whole* screen with color!! You can make the program repeat over and over again by adding a LOOP. This means that it goes in an endless circle from one line to another line that you send it to.
   Type this line and RUN the program again.

```
115 GOTO 40
RUN
```

NOTE: Remember RUN/STOP. Press RUN/STOP and RESTORE when you want to end the program run.

3. Practice:
   - **a.** Change the number of spaces between the " " in each line. See how different your patterns will be. Do it this way:
     Type LIST.
     Move the cursor up to each line and make the changes on the screen. Remember to press RETURN after you make the change on each line.
   - **b.** Add these lines:

```
35 CL=INT(8*RND(1))+1
38 ON CL GOTO
   40,50,60,70,80,90,100,119
```

     Change this line:

```
115 GOTO 35
```

     Now RUN the program again. You are giving the computer the chance to pick any of the eight colors *at random* (RND) by using the formula in line 35. See what patterns the computer makes by itself.

C. Backgrounds and Borders

Maybe you are tired of looking at a blue screen all the time. This program will show you what other possible color combinations look like. You will be using another computer code called POKE and PEEK. In this code, all sixteen colors are available in any combination. These are the numbers for the POKE color code:

0 black	6 blue	11 gray 1
1 white	7 yellow	12 gray 2
2 red	8 orange	13 light green
3 cyan	9 brown	14 light blue
4 purple	10 light red	15 gray 3

1. Type this program:

```
NEW
10 FOR BA=0 TO 15
20 FOR BO=0 TO 15
30 POKE 53280, BA
40 POKE 53281, BO
50 FOR X=1 TO 500:NEXT X
60 NEXT BO: NEXT BA
70 END
RUN
```

NOTE: We have let BA be the name for background and BO the name for border. 0 to 15 are the different numbers for colors. POKE 53280 is the location of a certain memory space for backgrounds. POKE 53281 is the location of a certain memory space for borders. Line 50 is a timing loop (remember how to slow down screen changes?).

2. Practice:
   - **a.** Type LIST and add these lines to your program:

This is the code to clear the screen.

```
25 PRINT CHR$(147)
26 PRINT "BORDER=";
   PEEK (53280) AND 15, "BACK-
   GROUND="; PEEK (53281) AND 15
RUN
```

   - **b.** Use the color chart to change the color numbers in lines 10 and 20

3. Questions:
   - **a.** Why did the printing disappear every so often? Hint: What color was the screen when it went blank?
   - **b.** Which colors were easiest to read? You can press RUN/STOP and make a note of the ones you might like to use for your own programs. Type CONT to *continue* viewing.

COMPUTER LICENSE TEST
Part 5

Color Me Purple

1. Write a program that will print your name all over the screen in a color other than blue. Run it for your teacher.

2. Write a program that will print your name or your school's name in columns down the screen, using a background and border of the *same color,* other than blue. Run it for your teacher.

3. Enter and run the program for *random automatic color bars* in which each different color bar is the exact width of the screen.

Student name____________________

Test date____________________

____

____

Date passed (100%)____________________

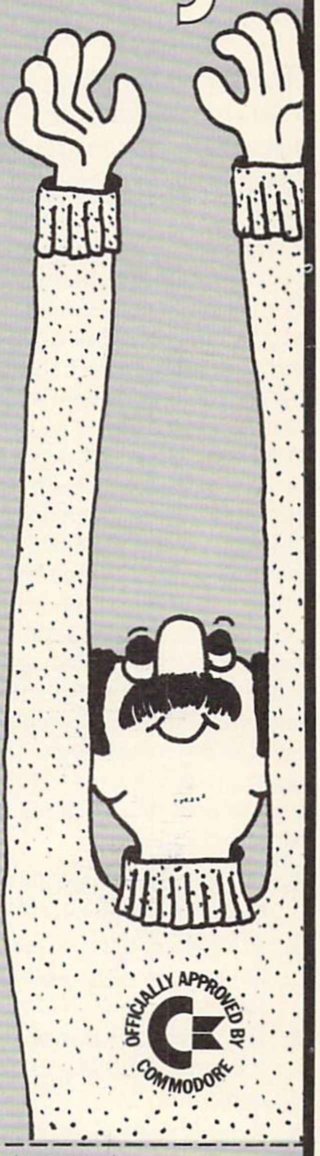
Approved by____________________

tired of software that doesn't work?

reach up & turn over to...

new leaf!™

All New Leaf Inc. products are created with our special committment to quality . . . because we really care. Unlike most software, even novice computer users will be able to enjoy all of the benefits, without any of the hassles!



BIORHYTHM+

Classic Biorhythm with audio feedback & compatability comparisons. Plus, ours will give you the option of a printed chart.*
Cassette only **14.50**

C*A*R*S

(Car Analysis Reporting System) Graph & monitor operating efficiency & cost of ownership including car insurance, payments, taxes & license fees as well as repairs. It distinguishes between operating & ownership costs. Again, ours will give you the option of printed reports.*

Cassette version	**24.50**
Disk version	**27.50**

ESP›Calc

(Electronic Spreadsheet Planning Calculator)
Now you can calculate any spreadsheet easier! Rows and columns are limited only by your machine's memory. An excellent tool for planning and repetitive calculations. You can even make printed copies of your spreadsheets.*

Cassette version	**43.50**
Disk version	**47.50**

*On any Commodore® compatible printer.

Each of our programs has been tested and re-tested over and over again. Our national testing network eliminates any problems BEFORE they reach you. Simply type Load & New Leaf's™ unique Self-starting format will do the rest.

Mastercard and VISA customers, please add 3%

MasterCard VISA

Product ordered__________
☐ VIC-20® ☐ Commodore 64®
☐ Disk ☐ Cassette
Amount enclosed __________
Name __________
Address __________
City______ State____ Zip____
Phone Number (   ) __________
MC or VISA # __________
Expiration date __________
Interbank # (MC) __________
Missouri residents add sales tax

new leaf inc.™
120 Lynnhaven • Belleville, IL 62223

Parent Computer Power

by Walter Herrala, Ph.D.
Principal, South Lyon, Michigan, Elementary School

How an elementary school on a limited budget uses "parent power" to enrich students' computing experience.

The kind of computer assisted instruction (CAI) children benefit from most, in our estimation, requires adult supervision. But when we started our computer project at the South Lyon elementary school we lacked the financial resources to pay for that supervision. As a result we were forced to turn to volunteer help—mainly parents—whom I trained to help the children get the most out of our PET computers. As it turned out, parent involvement works so well it has changed what started out as a handicap into a real advantage.

We all know principals are forever descrying the lack of parental involvement in school activities. The computer project is a perfect parent involvement project in our school for several reasons:

1. Parents are in a special area so teachers don't feel "invaded".
2. A volunteer parent is as likely to have the necessary skills as a teacher.
3. Involvement is during the day, so it doesn't conflict with parenting chores.
4. Parents perceive their involvement as very meaningful. They make an impact on the kinds of learning they like to see go on in schools, and they see the results every time they come in.
5. Their volunteer work does not threaten to expand beyond their wishes like some kinds of committee volunteerism.

The parents themselves also benefit in some ways from their work in our school. Computers have received such a build-up in the media that many people want to be involved, even though they may not be able to afford a computer. Parents in that position, who want to be up-to-date on what is happening in education, get a chance to find out what their children are learning and what they are going to be faced with for the rest of their lives. The user-friendly qualities of the PET make it possible for these untrained people to learn "on the job". There can also be spin-offs from computer projects, such as teacher computer classes, computer clubs, adult education classes and after-school classes, that provide further benefits to students, teachers, volunteers and the community at large.

A question that inevitably arises is whether we would prefer a trained professional if we could afford one. My answer is no. At this point such a person would be better used as a consultant to the staff and for teaching the children programming.

I should also mention, before I go into the details of how we set up our project, that I think it should be the responsibility of the school principal to oversee the setting up of a school computer project. Principals are the educational leaders, and should therefore provide educational leadership, rather than delegating that responsibility to staff members or outside specialists. It is also the easiest way to avoid the fragmentation of opinion that can occur when you must choose among many different brands of computers and a multiplicity of software.

However, in administering this project, I assume the software selected by the teacher is the best that could have been selected. The judgement on content is a professional one. I also assume that micros are efficient interactive devices if used with common sense judgement about student motivation and receptivity. Therefore, my job as the administrator is to maximize the amount of quality time the students spend interacting with the computers.

Hardware

The computer center uses an empty classroom adjacent to the office, library, teachers' lounge and front door. Carrels were built on top of sturdy school tables at a compromise height for all grades. The work was done by our building engineer. Each carrel is wired with a convenient switch so a single parent aide can switch eleven micros on in one turn.

Ten 4032 "fat forties" are networked to one 4040 dual disk drive and a printer. Since the parent aide alone is permitted access to the disk drive, or give permission to access it, the obvious problem of hanging up the network by attempting to access from two PETs at once is minimal. I should point out that the IEEE kind of network has a very low cost because the capacity is built right into the PET. Many administrators I know don't have the foggiest idea that this can be done for the price of a few inexpensive cables.

What an administrator buys in a disk drive, among other things, is the ability to have one person easily control several stations. If a cassette drive is used to load ten computers it will take 10x3 minutes—the exact amount we allot for a session! Cassette drives worked just fine for us in this same parent project until we got our disk unit, but that was because we did not change the programs nearly as often and we had only four computers. Now we can use three or four different programs a session, depending on the ability of the aide.

If the system gets hung-up we give ourselves one minute to solve the problem by working individual machines. If we do not succeed, we don't theorize. Off go all ten; on again; load in the DOS support, and we're back in business again.

Attracting Volunteers

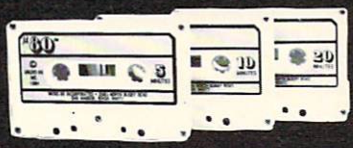
A letter was sent home with a date and time set for training parents who wanted to get involved with computers. The principal promised to teach them everything they needed to know about running our center. It was emphasized that they were needed and that they would be making a direct contribution to the learning process. We told them they would be asked to come for one half day per week and they would be working with a partner. Busy parents need to know that they can get in and out of the school on their terms. They definitely do not want to be in the center alone with ten children to check in and out.

Training

A single initial training session was conducted. In this session we used only terms that were necessary for them to know in order to function. They were to be the managers of the machines, but more importantly they were to sign each child into the center on his/her student user booklet. They were told that the teachers would be writing down each day's activities on the master planning schedule and that they need only follow it. We had each parent sit at the terminal and push all the keys to dispel any idea that they could hurt the machine. A very careful and repetitive lesson on loading a program from

COMPUTER CASSETTES

100% Error-Free • Fully Guaranteed



LENGTH	12 PACK	24 PACK
C-05	79¢	69¢
C-10	89¢	79¢
C-20	99¢	89¢
Boxes	26¢	21¢

UPS $3.00 Pkg. $18.00 Case

C-10's 39¢
(Min. 500 Case Lot)
w/labels ADD 4¢,
w/boxes ADD 13¢

FOR ORDERS ONLY
1-800-528-6050
Extension 3005

MICRO-80™ INC.
2665-P Busby Road
Oak Harbor, WA 98277

FOXSOFT™
Presents.....

SpryteByter™
For the Commodore 64™

The user affectionate sprite development program. Menu-driven, Mono/Multicolor sprites, joystick/keyboard, Tape/Disk. 20K w/FAST machine language routines. Over 60 commands: ROTATE (any angle 0-360), INVERT/OBVERT, SHIFTS, SYMMETRY, AND/OR, REVERSE, REVIEW, MOVIE (animation). Create and Edit up to 128 sprites per file. For programming efficiency and FUN!

PLUS
The Game Maker
Automatically prepares a base for game development-up to 32 sprites WITHOUT DATA STATEMENTS! Adds SPEED to your games. Saves memory too.

To order: Send check (U.S. Funds) $29.95 for Cassette, $34.95 for Disk. Dealer inquiries invited.

FOXSOFT™
P.O. Box 507
Deer Park, Texas 77536
(713) 473-6723

A Division of Foxfire Systems, Inc.

DEVELOP-20

FIVE POWERFUL SOFTWARE DEVELOPMENT TOOLS

Plus The Exciting New Book

INSIDE THE VIC

THE BOOK

A complete clear explanation of machine language, Assembly language, VIC 20 architecture, graphics, joystick and sound effect programming. Detailed step-by-step guide to the use of the development tools. How to combine BASIC and machine language, make auto-start cartridges, interface with the internal ROM-based programs of BASIC and the Kernal. Sample programs fully explained.

THE TOOLS

Assembler/Editor/Loader/Decoder/Monitor
Full-featured Assembler allows use of labels, comments and arithmetic expressions to create machine language programs. Create, save, modify Assembly language programs with the Editor. Load and link machine language modules with the Loader. Decode machine language back into assembly language for study or input to the Editor. Single-step program execution with the Monitor. Extended features combines Assembler/Editor for maximum ease of use.

ALL FOR $49.95 PLUS $2.00 POSTAGE AND HANDLING Standard version runs on any system with Datasette (5K and up) Add $5.00 for disk version, $5.00 for extended features (minimum 8K) Send check, M.O., VISA/MC ($2.00 S.C.) or specify C.O.D. (add $3.00) to:

French Silk smooth ware

P.O. Box 207, Cannon Falls, MN 55009

507-263-4821

VIC-20 is a registered TM of Commodore Business Machines Inc.

the disk drive was given. There were many questions and there still are.

Jump In and Swim

At what point are the aides properly trained? The answer is that learning about computers is like learning to debug a program. You know the thing should run but it doesn't. Attitude is important and parent aides will have a problem-solving attitude if you convey that attitude as the director. Convey the idea that everyone has problems. There is a period of confusion but it is followed by a gradual decline in the need for administrative attention. Make it clear that you want them to learn how to do it and they (almost always) will achieve independence.

Documentation

Administrators may be interested in documenting the exact accomplishments of their computer center. A center affords accurate and detailed accountability that is rare in the school setting. To aid in documentation, we created student user booklets using colored folder stock. The covers are festooned with attractive computer cartoons and terms and there is a place for a student name. They were professionally printed because the PTA wanted to pay for them. (Before their offer we simply ran the art work on mimeo and pasted it on stock folders.) Inside these folders there is a formatted place for entry of date, subject or program, number of problems worked, number correct and percent correct. At the end of the year these booklets are collected and a report is made on the total number of contacts as well as the number of problems done, etc. It can be made very clear that students are progressing if their levels are going up.

Master Schedule

Each computer station was given a number and a master planning sheet was created. Teachers signed their students up on the master sheet. Because this takes time on their part, it should be presented at a staff meeting. A good way to start if staff awareness is low is to buy a self-leveling program such as the Milliken math program, which automatically adjusts the level of difficulty upward. That way teachers can get a very quick idea of what they are signing up for without knowing a lot about programs available. Note that lack of teacher awareness of programs is not a roadblock in this plan.

Helpful Hints

1. Use duct tape to secure IEEE cables.
2. If students are allowed to access the disk drive (dd), they must have some token. Our token is a plastic dog. After access, the dog sits on the dd.
3. Write expectations for volunteers. Do this in detail. Volunteers want clear detailed directions. Include:
   - Clean up procedures
   - Power-up sequence
   - Care of disks, cassettes
   - Descriptions of loading
   - Location of manuals
   - Teacher expectations for selecting software
   - Student behavior expectations
   - Do nots = eating, smoking
4. Create a substitute list. Aides are asked to call their own subs.
5. Needless to say, back up everything. We keep cassette backups of our most widely used programs just in case our dd needs service.
6. Ask one of the aides to clean all units each week. We bought a cleaning kit for the dd.
7. Use the DOS command "Name" to load programs. We have the DOS program on a disk by itself. It is in an envelope stapled to the dd carrel. Before any loading is done, this program goes in each micro. The DOS will be operational until the computer is powered down.
8. Most problems are due to bad connections. Look for IEEE connections *partially* out or cockeyed. This problem is compounded if you try to use less IEEE cable than necessary.
9. Loaning computers sounds great but with networking it is not desirable. Too much reconnecting.
10. Make sure all computers in the network are on before attempting to LOAD.
11. A good tray-type disk container is best for aides who must quickly find and change programs. The price for these containers is outrageous. Make your own. Divide disks into content areas volunteers understand.
12. An updated list of programs is given to each teacher and aide. The aide must be able to quickly locate the disk. Use content areas.
13. Encourage volunteers to remain near the students and be helpful.
14. Volunteers need inservice and encouragement. Meet with them at least every six weeks.
15. Always be on the lookout for new volunteers. **C**

Microcomputers: Truly Child's Play

by Terry Anders

A child from the Brooklyn Park Kinder-Care learning center working on the new talking PET Commodore computer.

Having loaded in the software, the child is waiting to proceed with the program. Look at that anticipation!

A three-year-old loading a program, patiently waiting and then interacting with the computer? A vision of the future, perhaps? No, it is reality at Kinder-Care Learning Centers in Minneapolis, Houston and in the Montgomery, Alabama, area. Children from ages three to twelve are using Commodore microcomputers on a regular basis. For them it is but another new life skill to learn and one which they practice with glee!

Kinder-Care is the nation's largest proprietary child care provider, serving children from six weeks of age through age twelve. In June, 1982, ten learning centers in the Minneapolis area began using the Commodore PET microcomputer with natural voice programs. This pilot program was implemented by Donna Goff, Zone Manager in the Minneapolis area, as an addition to the GOAL (Growth Opportunities for Achievement and Learning) program. This is a program based upon children's natural curiosity and Ms. Goff felt that microcomputers were a natural addition to it. The computers make up yet another "discovery area" in the centers where children can explore the computer and use programs appropriate to their learning needs.

Ms. Goff believes that "Kinder-Care has introduced the learning tool of tomorrow, a 'friendly' microcomputer that talks to children on their own level and provides an opportunity for the teacher to help the child learn in an individualized setting".

Young children do not have to know how to read to use the computer successfully. In some situations they are able to interact with programs teaching concepts such as size, directionality, and quantity by using the natural voice and responding via a light pen (an electronic pen which, when pointed at the screen, can be "read" by the computer). As they develop symbol skills, they can begin to use the computer keyboard. What is important is that the children are *interacting* and getting immediate feedback on their work. The computer is infinitely patient and supportive of repeated attempts. The natural voice of the Kinder-Care-selected programs provide clear, understandable communication. It is a non-threatening, self-directed learning environment.

Kinder-Care believes that the computer-based learning can be used to support concepts taught in the classroom as well as present new material. Kinder-Care has an established curriculum for its national centers. Sanny Sue Holland, Corporate Director of Curriculum, reports that her staff is developing activities to provide maximum integration of classroom and computer experiences.

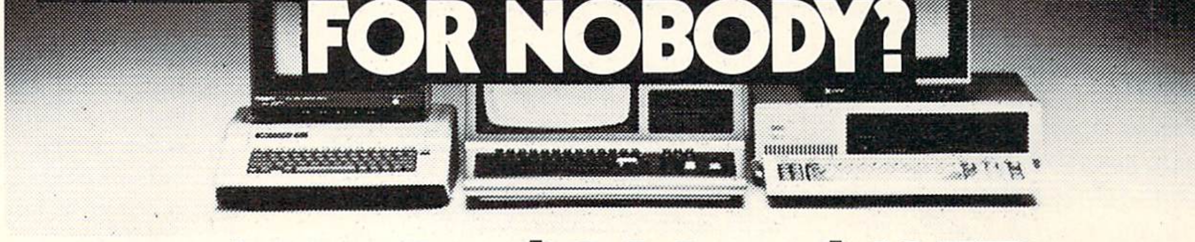
Children have responded with enthusiasm to the computers. At this stage in their lives they spend their days learning new skills and adapting to their environment. For them the computer is just another tool to master and set of skills to learn. They have no "computerphobia"—only openness to new situations. It is an ideal age for introduction to the computer which will certainly be with them throughout their lives.

In addition to the programs for the preschool and kindergarten children served by Kinder-Care, there is a Klubmates™ program for school-age children. These children also participate in the computer program and have the opportunity to earn a MicroKid™ badge indicating that they are "computer competent"!

In August, 1982, another test site was established in the Houston area. Computers were installed in 35 Houston area centers. As in the Minneapolis project, center directors and staff were thoroughly trained by both the software and hardware suppliers. In late fall it was decided to install yet another pilot test site in the Montgomery, Alabama, area, which is the home of Kinder-Care's corporate headquarters. At these sites the Commodore 64 microcomputers were installed in eleven centers. The 64 was selected because of its music and sound capabilities. Goals of these test sites include hardware and software evaluation, determination of the actual number of machines needed per center, the best configuration of these machines and obtaining yet another set of responses from children, parents and center staff.

Responses to date from children have been enthusiastic, and parents are pleased that their children are involved in preparing for the future while still very young. Gene Montgomery, Vice President Operations for Kinder-Care Learning Centers, Inc. states that "In this era of increased technology we are proud to help our preschool children become computer literate. This new approach to learning makes it possible for the child to understand the technology of the future while learning in an exciting environment.". **C**

IF PERSONAL COMPUTERS ARE FOR EVERYBODY, HOW COME THEY'RE PRICED FOR NOBODY?



$1395* APPLE® IIe 64K

$999* TRS-80® III 16K

$1355* IBM® PC 64K

That's a good question.

The Commodore 64 gives you 64K of memory. As much, or more, than any of its competitors.

But take a look at the difference in price.

The list price of the Commodore 64 is hundreds less than that of its nearest competitor. And almost a thousand less than its furthest one.

Stop in and see all the things the Commodore 64 is capable of doing.

You already know the one thing it won't do, which is cost you a lot of money.

commodore COMPUTER

In more homes than any other home computer.

*Manufacturers' suggested list prices as of March 20, 1983. Monitor included with TRS-80 III only. Apple is a registered trademark of Apple Computer, Inc. TRS-80 is a registered trademark of Tandy Corp. IBM is a registered trademark of International Business Machines Corp.

THE COMMODORE 64™

The Commodore 64 Trip Planner

by Charles Knight

Because of their length (about nine feet of program listing), we aren't able to print Chic's programs here. However, he says if you send him a blank disk, a self-addressed, stamped return mailer (sturdy) and $2.00 handling, he'll be glad to send you a copy. You can reach him at: 1732 County Road 995, Route 6, Ashland, OH 44805.

It was one of those things that sneaks up on you from time to time. You're not aware that it's there, or at least what its potential is, until suddenly it hits you like a downpour out of a clear sky when you are in the middle of the fifth fairway.

When the Commodore 64 was unpacked at our house in the fall of 1982 its purchase had really been as a sort of necessity and not for the purposes of recreation or experimentation. My wife is a data accounting teacher at a local vocational high school and while there had been a couple of Brand X business-type computer systems in place for a while, there were not enough machines to make much instruction practical. When the announcement came that more computers would soon be on the way it became evident that if she was going to be in a position to know much about teaching computer skills there would have to be a micro in our home. We investigated most of them and decided that the 64 with a companion disk drive and the 1525 printer would serve her needs.

The rains started falling on me when I realized that at 43 I wasn't too old to make the machine function in a productive manner and even do what I wanted it to do. Being a practical person (so my wife claims) posed a problem though because I didn't want to play games; I wanted the computer to do things for me that were practical—things that I could use later.

We as a family are avid RV'ers, belonging to a Columbus (Ohio) based chapter of the International Skamper Camping Club. When I say avid I mean that our latest travel trailer, purchased in the summer of 1980, has seen over 7,000 miles so far. It seemed natural then to mix our RV'ing and computing together and have the computer handle a batch of chores that to this point had been mostly done on paper with a calculator or log books.

Learning to program became the first chore, so with the help of the *Commodore 64 User's Guide*, the *Commodore 64 Programmer's Reference Manual*, and a book I would recommend highly to anyone wanting to teach themselves programming I was on my way. The book incidently is a high school text book on programming entitled *Structured BASIC* from the Southwestern Publishing Company of Cincinnati.

As confidence built I tried my first "practical" venture. We were planning a trip to Myrtle Beach, South Carolina, with the family and the camper this past spring. Anyone can read a map and total up the miles to see how far it is, but what I wanted to know was how long the drive would take. Simply taking the miles travelled and estimating the average speed you would maintain was not enough, because, while this gave you a rough idea, it did not take into consideration other factors such as time off the road for fuel stops, rest stops or meals or the different traffic conditions you would find enroute. Traffic conditions such as leaving I-77 at Charlotte, North Carolina, travelling through town on city streets and then travelling state and federal highways that didn't allow the steady speeds of the interstates.

To this end I wrote a program that, when combined with the printer, allowed you to input certain factors and then produce a printout to take with you that listed your intentions for stops and at the same time added time factors to your trip for those stops, thus giving you a more accurate picture of your total time enroute.

To use the program it is necessary to break your trip down into no more than ten segments in length with each segment having a minimum length of 20 miles and a maximum of 160 miles. You then pick a starting and ending point for each segment. Entering these into the computer then gives you back a recap of the information and when you enter your choice of three different average speeds it gives you the driving time for those speeds.

Then back to the keyboard where the program asks you to enter your intentions on each of the segments. For every fuel stop you enter, an additional 15 minutes is added. For rest stops again 15 minutes and for meals you are given a 35-minute cushion. In addition to this you are asked to enter, at each segment, any cities over 50 thousand where you will have to use an innerbelt system, or where you have to travel city streets. The innerbelt adds 30 minutes to your driving time, the city traffic an hour. The same is requested for cities from 20 to 50 thousand with innerbelts accounting for an additional ten minutes, the city streets for 20 minutes. Towns under 20 thousand and outerbelts, where the traffic usually moves along well, are not factored.

With this information in place you are given the total driving time for each of the three average speeds you selected and then you are asked to narrow that choice to one average speed. It does not have to be one of the previous three.

With this done the computer then continues its printout, giving you a record of each segment at the final average you chose and listing the segment points, the routes you indicated, the mileage of the segment, the number of rest, fuel and meal stops that you planned. It also tells you that considering all the factors you entered (meaning all these factors plus the city information), the drive from point A to point B at your average speed will be X number of hours expressed in tenths.

Each segment is listed separately until all ten have been recapped. Then you are given the total time enroute and a comparison graph of the segments. If, as in the case of our camping club, the trip involves more than one vehicle, you are given the option of printing additional copies of the segment-by-segment portion. When you are finished you have a very personal, highly informative trip schedule.

The real test for the program came when we set out for Myrtle Beach and, except for a couple of unplanned rest stops, too much coffee (an unfactored item), the times were near enough to call the program a success.

An interesting sidelight to the program is that word got around about what I was doing and someone at the office asked me for a copy for them to use for a different trip. This caused a small problem because I had to explain what kind of information I would need to make their schedule. As the explanation became rather lengthy, I turned to a new task and created a questionnaire that asks for all the information needed to make the program work. It saves a lot of time in the way of explanations and I plan to use it myself in the future as a guide to what I want to do when on other trips.

With the success of the trip planner I jumped in with both feet and decided that all the records associated with our camping could be computerized just as easily, giving me the ease of recall that I would need if they were to be practical. These programs, which center around reading data statements into an array, are proving to be worth their weight in gold. They have been so easy to use that the original ideas have been expanded to the point that I now have every piece of information that I need during the camping season at my fingertips almost instantly.

The first of the programs to be created was CAMPER STATS. A program where I could readily keep track of such varied items as annual and total mileages, campouts we have been to with the club, family camping ventures on our own, expenses for the camper, very important serial numbers and phone numbers for suppliers (of appliances, our camper itself and the like).

The menu for CAMPER STATS provides several options including mileage totals for each year since the camper was delivered, a running total of the mileage to date. Both listings give me the date, destination and the miles travelled. The second function allows me to examine for each year, or as a total to date, all the campouts we have attended with the club. This printout gives me the dates, locations and mileage. The same is true for camping activities as a family outside of the club structure.

Camper expenses are another option that I can recall, giving me the reason for the expense, the cost, where purchased and the date. These expenses are recallable by year and as a total since the camper was new. It allows me to check back on when we did what and what item, wheel bearings as an example, may be coming up for main-

COW BAY COMPUTING
has a lot to offer you and your
PET/CBM/COMMODORE 64

SOFTWARE FOR THE CLASSROOM

o The Pet Professor $499.00

o With Management System $649.00

A total arithmetic package with step-by-step instruction.77 programs in addition, subtraction, multiplication and division on cassettes or disks. Ask for a sample.

Specify disk drive and computer configuration.

WORKBOOKS FOR COMPUTER LITERACY

o Feed Me, I'm Your PET $5.95

o Looking Good With Your PET $5.95

o Teacher's PET $5.00

STUDENT SCHEDULING SYSTEM

o Fast, accurate, easy-to-use and less expensive than current methods.

o Uses a 8032 computer and 8050 disk drive

o 90 Day Money Back Guarantee to prove that it can be done.

COW BAY COMPUTING
P.O.Box 515
Manhasset, N.Y. 11030
(516) 365-4423

tenance again.

Finally the menu offers a section on vital statistics and data input format. In the vital statistics section are stored model and serial numbers for the camper and its appliances, phone numbers for suppliers and dealers and vehicle licensing information. The final choice is a section that gives me examples of the data lines and how each is filled out for the varied items. It tells me at which line number the data statements begin and how each is formatted along with notes that might require a special formatting in certain situations.

The key to the use of the program is first reading the data into an array and secondly coding the data statements by number, such as DATA1, DATA2 and so forth. It makes sorting the information out much easier and faster.

In the second program the same basic format was used. It reads data into an array and uses numbered data lines to key in a particular function. The program, CLUB-CAMPS-ROUTE, does not have as full a menu as CAMPER STATS, at least not at this point, but it more than serves the purpose for which it was intended. At the top of the menu is a listing of the members of the camping club, which is intended not as a mailing list but rather to provide quick access to addresses and telephone numbers as well as club unit members. You may search this section for a particular individual or recall the entire list.

Have you ever taken a particular trip, and when you decided to go back again to the same place, have trouble remembering the routing you used? The second section of the menu allows you to recall trip routings by entering the destination point or by calling up the entire listing.

Campground listings compose the major section of the program and are the listings that will in all probability prove to be the most useful. Here, entered in data lines, are the names, addresses and telephone numbers of campgrounds that are repeatedly used or are favorites. They may be recalled by entering the name of the town where the campground is located, or, should you not remember the town,

```
YOUR   PERSONAL   TRIP   SCHEDULE

-----------------------------------------------------------------

ASHLAND - MYRTLE BEACH

THE LENGTH OF EACH SEGMENT IS:
SEGMENT 1 - ASHLAND TO STRASBURG.......US 250 =  52  MILES

SEGMENT 2 - STRASBURG TO MARIETTA.......I-77 =  91  MILES

SEGMENT 3 - MARIETTA TO RIPLEY.........I-77 =  50  MILES

SEGMENT 4 - RIPLEY TO PRINCETON.....I-77/W.VA.TP. =  124  MILES

SEGMENT 5 - PRINCETON TO I-81..........I-77 =  36  MILES

SEGMENT 6 - I-81 TO I-40.............I-77 =  97  MILES

SEGMENT 7 - I-40 TO CHARLOTTE..........I-77 =  38  MILES

SEGMENT 8 - CHARLOTTE TO WADESBORO.....US 74 =  59  MILES

SEGMENT 9 - WADESBORO TO I-95...US 52/SR 9/SR 38 =  60  MILES

SEGMENT 10 - I-95 TO MYRTLE BEACH....SR 38/US 501 =  65  MILES
```

entering the abbreviation for the state and recalling all listed campgrounds in that state. If you want the total listing, that is also available. An added feature of this section is the listing of favorite campsite numbers so that when you make reservations you can request the same campsite.

Data input information is also a feature of this program as well. A menu choice allows you to find the beginning of data lines by number, gives you examples of each type of input and also gives you special notes that are peculiar to particular entries.

Both of the programs offer you a choice of screen or printer display. In some cases you can have both, in others you choose between the CRT or the printer. Say, for instance you want to return to that favorite campground next month. You can punch it up on the screen and get the phone number and preferred site number so that you can make reservations. Then before you leave you can printout that information for the office, relatives or anyone else who might need to find you in an emergency. Or, in the case of data input info, you might need just a printout to refer to as you enter new data into the program.

All-in-all there is no doubt that my 64 and the RV and travel programs we created together are about as indispensible to me as the trailer hitch on the car. Surely if it had not been for the ease of operation that the 64 provides over some other computers I have seen the programs would not have been as easy as they were to create, edit and debug.

As to the original reason for our 64 purchase, well, my wife claims she doesn't get enough time on our 64, and you know something? She's right. So are our two sons, who have the same complaint. If this keeps up we easily could become a two 64 family. And who says computers are toys? Toys, heck, they are a practical necessity in this day and age. If you want to play games on your 64 that's fine with me, but I'd rather camp with mine. **C**

Following is a sample of what Chic's program gives you.

```
WE ALSO KNOW THE TOTAL MILEAGE = 672

WE KNOW THAT THE BASIC TOTAL TIME IS:
1.... 45 MPH = 14.93 HOURS

2.... 50 MPH = 13.44 HOURS

3.... 55 MPH = 12.22 HOURS
NOW WE ARE READY TO TIME EACH LEG
AND DO IT EXACTLY!!!!!!!!!!!!!!!!
------------------------------------------------------------------------

I WILL ADD TO ADJUST TIMES -

1. FUEL + 15 MIN.,MEALS + 35 MIN.,REST STOPS + 15 MIN.

2. CITIES OVER 50 THOUSAND WITH INNERBELTS WILL BE + 30 MIN.

3. CITIES OVER 50 THOUSAND WITHOUT INNERBELTS + 1 HR:OUTERBELTS
   + 0 TIME'

4. CITIES FROM 20 TO 50 THOUSAND W/INNERBELTS + 10 MIN:WITHOUT
   + 20 MIN
```

```
5. TOWNS UNDER 20 THOUSAND ARE NOT FACTORED.
-----------------------------------------------------------------------

++++++++++++++++++++++++++++++++++++++++

TOTAL MILES - ASHLAND - MYRTLE BEACH IS  672  MILES

TOTAL ADJUSTED TIME ENROUTE AT 45 MPH IS 20.71 HOURS

TOTAL ADJUSTED TIME ENROUTE AT 50 MPH IS 19.22 HOURS

TOTAL ADJUSTED TIME ENROUTE AT 55 MPH IS 17.99 HOURS

ON YOUR QUESTIONNAIRE YOU CHOSE 50 MPH.
USING YOUR CHOICE OF 50 MPH I WILL RECAP EVERYTHING -
TAKING ALL THE FACTORS PROVIDED AND ENTERED INTO CONSIDERATION.

HERE IS YOUR PERSONAL TRIP SCHEDULE, MADE JUST FOR THE KNIGHTS
-----------------------------------------------------------------------
ASHLAND - MYRTLE BEACH =  672  TOTAL MILES
-----------------------------------------------------------------------

SEGMENT ONE
ASHLAND TO STRASBURG........US 250 EQUALS  52  MILES

MEAL STOPS -  0
FUEL STOPS -  0
REST STOPS -  0

CONSIDERING ALL THE FACTORS YOU ENTERED EARLIER IN THE PROGRAM
ASHLAND TO STRASBURG........US 250
AT  50  MPH EQUALS  1.22  HOURS
-----------------------------------------------------------------------

SEGMENT TWO
STRASBURG TO MARIETTA........I-77 EQUALS  91  MILES

MEAL STOPS -  0
FUEL STOPS -  0
REST STOPS -  2

CONSIDERING ALL THE FACTORS YOU ENTERED EARLIER
STRASBURG TO MARIETTA........I-77
AT  50  MPH=  2.22  HOURS
```

```
-----------------------------------------------------------------------

SEGMENT THREE
MARIETTA TO RIPLEY.........I-77 EQUALS  50  MILES

MEAL STOPS -  1
FUEL STOPS -  1
REST STOPS -  0

CONSIDERING ALL THE FACTORS YOU ENTERED EARLIER
MARIETTA TO RIPLEY.........I-77
AT  50  MPH=  1.8  HOURS
-----------------------------------------------------------------------

SEGMENT FOUR
RIPLEY TO PRINCETON.......I-77/W.VA.TP. EQUALS  124  MILES

MEAL STOPS -  1
FUEL STOPS -  1
REST STOPS -  1

CONSIDERING ALL THE FACTORS YOU ENTERED EARLIER
RIPLEY TO PRINCETON.......I-77/W.VA.TP.
AT  50  MPH=  3.48  HOURS
-----------------------------------------------------------------------

SEGMENT FIVE
PRINCETON TO I-81...........I-77 EQUALS  36  MILES

MEAL STOPS -  0
FUEL STOPS -  0
REST STOPS -  1

CONSIDERING ALL THE FACTORS YOU ENTERED EARLIER
PRINCETON TO I-81...........I-77
AT  50  MPH=  .92  HOURS
-----------------------------------------------------------------------

SEGMENT SIX
I-81 TO I-40..............I-77 EQUALS  97  MILES

MEAL STOPS -  0
FUEL STOPS -  1
REST STOPS -  2
```

```
CONSIDERING ALL THE FACTORS YOU ENTERED EARLIER
I-81 TO I-40.............I-77
AT  50  MPH=  2.54  HOURS

--------------------------------------------------------------------------------

SEGMENT SEVEN
I-40 TO CHARLOTTE..........I-77 EQUALS  38  MILES

MEAL STOPS -  0
FUEL STOPS -  0
REST STOPS -  1

CONSIDERING ALL THE FACTORS YOU ENTERED EARLIER
I-40 TO CHARLOTTE..........I-77
AT  50  MPH=  .96  HOURS

--------------------------------------------------------------------------------

SEGMENT EIGHT
CHARLOTTE TO WADESBORO......US 74 EQUALS  59  MILES

MEAL STOPS -  1
FUEL STOPS -  1
REST STOPS -  0

CONSIDERING ALL THE FACTORS YOU ENTERED EARLIER
CHARLOTTE TO WADESBORO......US 74
AT  50  MPH=  2.98  HOURS

--------------------------------------------------------------------------------

SEGMENT NINE
WADESBORO TO I-95....US 52/SR 9/SR 38 EQUALS  60  MILES

MEAL STOPS -  0
FUEL STOPS -  1
REST STOPS -  1

CONSIDERING ALL THE FACTORS YOU ENTERED EARLIER
WADESBORO TO I-95....US 52/SR 9/SR 38
AT  50  MPH=  1.6  HOURS

--------------------------------------------------------------------------------

SEGMENT TEN
I-95 TO MYRTLE BEACH....SR 38/US 501 EQUALS  65  MILES
```

```
MEAL STOPS -  0
FUEL STOPS -  0
REST STOPS -  1

CONSIDERING ALL THE FACTORS YOU ENTERED EARLIER
I-95 TO MYRTLE BEACH....SR 38/US 501
AT  50  MPH=  1.5  HOURS
---------------------------------
TOTAL TIME ENOUTE AT  50  MPH IS
 18  HOURS
--------------------------
--------------------------------------------------------------------
COMPARISON OF MILES PER SEGMENT

10 !   40 !       80 !       120 !      160 !
   !      !          !           !          !
SEGMENT 1
▒▒▒▒▒▒▒▒▒▒▒▒▒▒
SEGMENT 2
▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
SEGMENT 3
▒▒▒▒▒▒▒▒▒▒▒▒
SEGMENT 4
▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
SEGMENT 5
▒▒▒▒▒▒▒▒▒
SEGMENT 6
▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
SEGMENT 7
▒▒▒▒▒▒▒▒▒
SEGMENT 8
▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
SEGMENT 9
▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
SEGMENT 10
▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
--------------------------------------------------------------

YOU'RE ALL SET, THE KNIGHTS

WISH I WERE GOING ALONG

H A V E   A   S A F E   T R I P   !
```

No List/Save

by David Williams
Reprinted with permission from *InfoAge,* January, 1983

During a recent meeting of the Toronto PET Users' Group, a member asked if there was any way to prevent users from LISTing programs. It turned out that he was a teacher whose "students" had the annoying habit of LISTing and deliberately altering programs, and then of asking him to diagnose and cure the problem. In answer to the question, I demonstrated the following technique, which provides a slight deterrent to this behaviour.

If the program already has a line numbered zero, change its number to something else. Then write the following line at the start of the program:
0 REMXXXXXXX
It is important that there should be seven "X"s after the REM. Then carry out the following instructions in direct mode.

PET/CBM:

```
FOR I=1030 TO 1035 : POKE I,20 : NEXT :
    POKE 1036,219
```

Commodore 64:

```
FOR I=2054 TO 2059 : POKE I,20 : NEXT :
    POKE 2060,204
```

VIC 20:

```
FOR I=4105 TO 4110 : POKE I,20 : NEXT :
    POKE 4111,204
```

Now enter the instruction "LIST". The PET will respond with the SYNTAX ERROR message and will refuse to list the program. However the program will still RUN properly. It can also be SAVEd in this form, so that it will automatically be protected after it has been re-LOADED.

The degree of protection offered by this method is, however, very slight. The direct-mode POKE instructions change the "X"s in the REM at line zero to characters which mess up the LIST routine when it tries to decipher this line. However, if the instruction "LIST-1" is entered, line zero will be skipped, and the rest of the program will be listed normally. It is possible to put additional lines of this type into the program, but finding them in memory in order to address the POKEs properly is a problem, and the additional protection is not great.

I was astonished at the amount of interest this little demonstration generated. It seems that many teachers suffer from the activities of students who enjoy messing up programs. They also, it turns out, frequently find the kids making illicit copies of copyright software. Preventing these two activities has become a major concern in computer-equipped classrooms.

I therefore wrote the following little program which provides a somewhat better degree of protection than does the technique described above. It works by patching a machine language routine into the PET's operating system. This looks for LIST, SAVE and DSAVE instructions and refuses to carry these out. All other operations of the computer are unaffected. To use this routine, simply load and run the program. It will ask whether you want "reset protection." If you answer in the affirmative, the effect of any of the forbidden instructions (or of an attempt to break to the machine language monitor) will be to make the PET reset itself, completely erasing the programs from its memory. This is an effective way to prevent LISTing or copying! If, however, you decide that this is too extreme, answer "no" to the question. The forbidden instructions will then only make the words "don't do that!" appear on the screen, nothing more. You can then load whatever program you wish your students to use, and they can do so without easily being able to either LIST or SAVE it.

No system such as this, which is purely software-based, can provide a completely impregnable degree of protection. It is always possible to reverse the changes which it makes to the operating system. The simplest way to do this is simply to turn the machine off, then on again, and to re-LOAD the program which you were trying to protect. If however, you take steps to prevent this, such as by removing the tape or disk copy of the program, the task of disabling the LIST/SAVE prevention routine without resetting the machine becomes one which requires some fairly detailed knowledge of the PET's inner workings. I could certainly do it, but I'm not about to publish how! **C**

No List/No Save for PET/CBM

```
0 REM 'NO LIST/SAVE', DAVID WILLIAMS, 1982
1 :
2 REM PROGRAM DISABLES 'LIST', 'SAVE' AND 'DSAVE' COMMANDS.
3 REM THIS SHOULD BE USEFUL FOR COPYRIGHT PROTECTION IN SCHOOL ENVIRONMENTS ETC
4 REM IF 'RESET PROTECTION IS CHOSEN, ANY ATTEMPT TO USE ANY OF THESE COMMANDS
5 REM (OR TO BREAK TO ML MONITOR) WILL CAUSE PET TO RESET, ERASING ALL PROGRAMS
6 REM (INCLUDING THIS ONE) FROM MEMORY.
7 REM MINOR LIMITATION: PROBLEMS MAY OCCUR IF SHIFTED CHARACTERS
8 REM APPEAR IN 'REM' STATEMENTS IN PROGRAM BEING PROTECTED.
9 :
10 S=909:F=987
20 FORI=STOF:READA:POKEI,A:C=C+A:NEXT:IFC<>9398THENPRINT:PRINT"DATA ERROR":STOP
30 SYSS
40 PRINT:PRINT"DO YOU WANT RESET PROTECTION? (Y/N) ";
50 GETG$:IFG$<>"Y"ANDG$<>"N"THEN50
60 PRINT"▓";G$:IFG$="N"THEN90
70 POKE146,PEEK(65532):POKE147,PEEK(65533)
80 POKE947,0
90 REM
1000 DATA169,76,133,112,169,154,133,113,169,3,133,114,96,230,119,208
1001 DATA2,230,120,72,152,72,160,0,177,119,201,155,240,8,201,148
1002 DATA240,4,201,213,208,19,169,128,145,119,152,200,145,119,160,16
1003 DATA185,203,3,32,210,255,136,208,247,104,168,104,76,118,0,13
1004 DATA33,84,65,72,84,32,79,68,32,84,39,78,79,68,13
```

To use the program on the VIC 20 and Commodore 64,
change lines 70-1004 as follows:

```
70 POKE790,PEEK(65532):POKE791,PEEK(65533)
80 POKE947,0
90 REM
1000 DATA169,76,133,115,169,154,133,116,169,3,133,117,96,230,122,208
1001 DATA2,230,123,72,152,72,160,0,177,122,201,155,240,8,201,148
1002 DATA240,4,201,213,208,19,169,143,145,122,152,200,145,119,160,16
1003 DATA185,203,3,32,210,255,136,208,247,104,168,104,76,121,0,13
1004 DATA33,84,65,72,84,32,79,68,32,84,39,78,79,68,13

READY.
```

Dollars and Cents Make Sense

by Joe Rotello

Line up decimals automatically when you're doing money calculations, and avoid making costly errors. This program is set up for the Commodore 64 but can be used with any Commodore computer by making the modifications mentioned in Joe's line-by-line explanation.

Among the myriad applications of your Commodore computer are programs that call on the user to input numeric data in the form of "dollars and cents", for example, 123.45 or .32.

One of the features you might need as a user of such a program is the ability to enter many financial figures rapidly. Another might be the ability to print those figures to the video screen or printer in a justified manner, that is, where the cents portion of the numbers all line up to the right of the decimal point, something like this:

```
 156.23
8956.12
   7.94
    .02
```

The program we are about to describe is one method for performing the following functions:

1) Allowing number entry in dollar-and-cents format without having to enter the decimal point. For example, 234.45 would simply be entered as 23445. The computer program recognizes that the user means "two hundred thirty-four dollars (and) forty-five cents".

2) Expanding on (1) above, the user may enter "2" or "02" and the program still recognizes that the user means "two cents" and prints out ".02" to the video screen or printer. Further, the user may enter an even dollar value, for example "100", and the program will correctly recognize the ".00" so the printout will appear as "1.00". (BASIC is notorious for wanting to forget about the ".00" and loves to display even-dollar values as "1", which will generally mess up not only printouts but your mind as well.)

3) The program will correctly justify any number when printing it either to the video screen or to any (we said any) CBM or other compatible printer. At this time, the maximum figure we can work with is eight actual digits, or 999999.99.

4) The program allows negative amounts, for example −124.03, to be entered. At this time, our program does not add or subtract these numbers, but you can modify the routines for inclusion into your own programs.

What follows is not the ultimate answer, but it does serve to show how to accomplish both "decimal-less" entry and number justification at the same time. The program is more of a tutorial so you can take the routines presented here and mold them to your own needs and programming style. The end result is that our "dollars-and-cents" financial data entry is greatly speeded up now that we don't have to worry about that !##%"$! decimal point every time we enter a figure.

The code is about 80 lines in length, half of those being REMark statements that aid us in seeing how the code works. For this first time, enter the program as it is presented. Later on, you can remove the REMS and put more multiple statements on one line in order to shorten the code and speed things up in BASIC.

The justification feature permits a very neat and organized look in your reports and listings, on video or printer. Those who are blessed with PETSpeed™ can compile the program for even faster results. (If you don't have PETSpeed yet, shame on you!! Go out and buy a copy and then finish this article!)

Some information on a few important lines is in order:

190 Variable T sets the tab for the video AND the printer output.

230 Variable LE sets the maximum digits allowed by the program. You could change LE to allow for custom program uses.

350 Those who use 8000/9000 series CBMs may want to take advantage of the available CHR$ function to erase a line.

370 This second method of erasing a line works on all CBM products. If you want, use this method instead of line 350 (but not both at the same time!). The line states "[home cursor][4 cursor down](27 spaces) [1 cursor up]". VIC 20 users will have to make allowances for the 22-character VIC screen width.

530 The final product, z$ contains the justified number in string format, printed variable T number of spaces on the video screen or printer.

690 Modified "Universal Data Entry" routine (see March, 1983, *Commodore Magazine*). Note that the routine allows only numbers and a possible "—" to be entered. If the user inputs letters or decimal points, they are ignored. Use the [INST/DEL] key to delete or correct the entry *before* pressing the [RETURN] key. **C**

```
10 REM *******************************************
20 REM *                                         *
30 REM * PROGRAM TO ENTER DOLLAR AMOUNT WITHOUT *
40 REM * NEED TO KEY IN THE DECIMAL POINT        *
50 REM *                                         *
60 REM * PROGRAM ALSO JUSTIFIES NUMBERS FOR      *
70 REM * PRINTING TO EITHER VIDEO OR PRINTER     *
80 REM *                                         *
90 REM * PROGRAM ALLOWS NEGATIVE NUMBERS TO      *
100 REM* BE REPRESENTED AS ''-XX''               *
110 REM*                                         *
120 REM*   J. ROTELLO 03 12 83                   *
130 REM*  COMPU SYSTEMS MANAGEMENT               *
140 REM*                                         *
150 REM******************************************
160 REM
170 REM
180 REM    SET PRINT TAB POSITION
190 T=25
200 REM  SET CONSTANTS
210 V$(0)=CHR$(175):V$(1)=" ":FORZ=1TO25:DZ$=DZ$+"▌":NEXT
220 REM  SET MAX ENTRY, 8 DIGITS (99999.99)
230 LE=8
250 REM  SELECT DISPLAY DEVICE
260 PRINT"▧ V ■IDEO OR ▧ P ■RINTER ?"
270 GETW$:IFW$=""THEN270
280 IFW$="V"THENX=3:GOTO310
290 IFW$="P"THENX=4:GOTO310
300 GOTO270
310 PRINT"⌐":W$=""
320 PRINT"■ENTER AMOUNT IN CENTS"
330 PRINT"ENTER <RETURN> TO END"
340 REM  ERASE LINE (8000/9000 COMMAND)
350 REM  PRINT"■■■■"CHR$(22);
360 REM  ERASE LINE, PRINT 27 SPACES (OTHER PETS, ETC)
370 PRINT"■■■■                           ⌐"
```

INNOVATIVE ORGANIZERS, INC.

For Commodore 64™ & VIC-20™

WELCOME, to the family of **INNOVATIVE ORGANIZERS.** These **ATTRACTIVE** power organizer stands, made of **ABS PLASTICS, U.L. RATED,** provide an opportunity for a more **PROFESSIONAL** looking computer installation, both at **HOME** and in the **WORK PLACE.**

There is **AMPLE** room under the organizer to place all the computer accessories: **POWER SUPPLY, MODEM, GAME CARTRIDGES, MEMORY EXPANSION,** and **MUCH MORE.**

Now, let's examine how these organizers work. There are 3 CRT-Models.

CRT-1 holds a Monitor or T.V. Set and a VIC 20 or 64. All wires & cartridges fit neatly under the organizer.

CRT-2 same as CRT-1, but also has: **ONE-STEP MASTER SWITCH** (permits 4 pieces of hardware to be turned on and off from one switch, plus circuit protection, incoming power fused with 6 AMP Fuse).

CRT-3 same as CRT-2, but also has: **SURGE & SPIKE PROTECTION** built in.

P-D-1: **HOLDS 1525 PRINTER & 1541 DISK DRIVE.** All wires & cartridges fit neatly under the organizer.

Ask about our Power Lock Switch & Power Portable Computer Fan & 2 Outlet Surge & Spike Protection.

Also: At present New Models are being developed for Commodore P-500 and B-500 Series.

I.O. INNOVATIVE ORGANIZERS YOUR ALTERNATIVE INNOVATIVE ORGANIZERS

VIC 20 and Commodore 64 are trademarks of Commodore Electronics, L.T.D.

P.O. Box 108, Markleville, IN 46056

CRT-1	$ 35.95
CRT-2	$ 59.95
CRT-3	$ 79.95
P-D-1	$ 38.95

Plus Shipping

CALL TO ORDER
317/533-2460

Trouble — DIAL
317/646-8943

Charge it on Visa or Master Card

Dealer Inquiries Invited

```
380 REM  NULL PREVIOUS ENTRY, ENTER YOUR NUMBER
390 A$="":B$="":PRINT"INPUT AMOUNT.......";
400 ZM=LE:GOSUB690:IFBB$=""THENPRINT"T]":GOTO420
410 A$=BB$:Z1=0:BB=0:BB$="":ZM=0:GOTO450
420 PRINT"[illegible] OK - STOPPED!":STOP
430 REM  DEVELOP B$
440 REM  EVALUATE, IF ONLY CENTS THEN FORMAT ACCORDINGLY
450 S=LEN(A$):IFS=1THENB$=".0"+A$:GOTO480
460 REM  EVALUATE, IF ''DOLLARS & CENTS'' FORMAT ACCORDINGLY
470 B$=LEFT$(A$,S-2)+"."+RIGHT$(A$,2)
480 PRINT"[illegible]"
490 OPEN6,X
500 REM  GO THRU ROUTINES & LOOP BACK FOR MORE
510 GOSUB550:GOSUB530:CLOSE6:GOTO320
520 REM  PRINT TO DEVICE SELECTED AT START OF PRG
530 PRINT#6,TAB(T)Z$:RETURN
540 REM  SET ROUTINE CONSTANTS
550 B1$=" ":D$="$":Z1$="0":ZZ$=".00":D=10:DD=100:Z$=""
560 REM  CHANGE NUMBER INTO CENTS
570 X1=VAL(B$):KS=INT((X1+5E-03)*DD):KT=KS/DD
580 K$=STR$(KT):IFX1<0THEN620
590 REM  STRIP SIGN OFF NUMBER
600 K$=RIGHT$(K$,LEN(K$)-1)
610 REM EVALUATE, TACK ON ".00" WHERE NEEDED
620 KU=INT(KS/D):KV=INT(KS/DD)
630 IFINT(KS-DD*KV)=0THENK$=K$+ZZ$:GOTO650
640 IFINT(KS-D*KU)=0THENK$=K$+Z1$
650 IFLEN(K$)<LE+1THENK$=B1$+K$:GOTO650
660 REM  REBUILD NUMBER SO WE CAN PRINT IT
670 Z$=D$+K$:RETURN
680 REM  MODIFIED UNIVERSAL DATA ENTRY ROUTINE
690 BB$="":FORZ=0TO9:GETZ1$:NEXT:PRINTCHR$(7);
700 PRINTV$(-V)"▮";:FORZ=0TO9:GETR1$:IFR1$=""THENNEXT:V=NOTV:GOTO700
710 Z1=0:IFR1$>""THENZ1=ASC(R1$)AND127:IFZ1=13THENPRINT" ":RETURN
720 REM  ALLOW ONLY NUMBERS INPUT
730 IF(Z1<48ORZ1>57)ANDZ1<>20ANDZ1<>45ORZ1=34ORZ1=46THEN700
740 BB=LEN(BB$):IFZ1=20ANDBB<1THEN700
750 IFZ1=20ANDBB=1THENPRINT" ■ ▮▮ ▮";:GOTO690
760 IFZ1=20THENBB$=LEFT$(BB$,BB-1):PRINT" ■ ▮▮ ▮";:GOTO700
770 IFBB=ZMTHEN700
780 BB$=BB$+R1$:PRINTR1$;
790 GOTO700
```

Getting the Most Out of (And Into) Your Disk Drive

by John Heilborn

Part 2

Disk Operating System (DOS) commands that help you locate data. Part 1 appeared last issue.

In the last article we covered the way in which the disk drive stores data onto the diskette and how to read sectors (blocks) of data and display them on the video screen (your television). This article will show you how the disk finds the information that was put on the diskette and how you can write data onto the diskette directly; without using the SAVE or typical data storage commands.

The Disk Format

When you buy a new diskette, it is totally blank, like a piece of paper. Before you can write onto or read from a diskette, you need to FORMAT the diskette. Formatting is a process that divides the diskette up into separate tracks and sectors. These tracks and sectors are where the data will be stored. The FORMAT program divides a diskette into thirty-five tracks with up to twenty-one sectors per track. All-in-all, a single diskette has more than six hundred sectors, each of which can hold up to two hundred and fifty six bytes of data. That comes to more than 175,000 bytes of data on a diskette.

The Organization of Data

Finding a specific part of all that data would be an enormous task if the disk drive didn't have some convenient method of looking for your information. To help find a particular file on the diskette, the disk drive maintains a directory on track eighteen of each diskette. The directory contains the name of every program on the diskette and its location by track and sector. Take a look at the first sector of the directory by entering this short program:

```
10 OPEN15,8,15
20 OPEN2,8,4,"#"
30 PRINT#15,"B-R:"4;0;18;1
40 GET#2,A$
50 PRINTA$
60 IFST=0THEN40
70 CLOSE2
80 CLOSE15
```

By using different numbers instead of 18 and 1 in line 30, you can read any track or sector on the diskette with this routine.

Writing to the Diskette

In addition to the diskette directory, the disk drive also maintains a file that indicates which sectors have data in them and which sectors are available for data storage. The file that contains this information is called the Block Availability Map (BAM). Normally, whenever you write data onto the diskette, the program that writes the data (which is built into the computer) looks at the BAM and finds the next available free sector. Unfortunately, none of the functions covered in this article support that software, so we need to read and update the BAM manually. The commands that do these functions are Block-Free and Block-Allocate.

Looking for Space on the Diskette

Let's assume that you have a block of data (256 bytes) that you want to save on the diskette but you want to be sure that you don't write over some existing data. To make sure that a given sector is available, all you need to do is check the BAM with the Block-Allocate command. This command will do one of two things when you run it. It will either allocate that sector (mark the sector as unavailable in the BAM) or (if the sector is already full) it will tell you the location of the next available track and sector. The information about the next track and sector as well as whether the sector is full or not is transmitted through the error channel.

What (you may be asking) is the error channel? The error channel (also known as the command channel) gives you information about certain error conditions that you may optionally read when you are accessing the disk drive. For example, if you try to write to a diskette that is write-protected, the diskette activity light will start to flash and the file will not be written. Sometimes, however, you will get a flashing light and you won't know why. If you read the disk error channel you will be able to read the error message generated by the disk drive itself (it knows what went wrong). In this case, the error message would be WRITE PROTECT ON. To read the error channel use the following lines:

```
10 OPEN15,8,15
20 INPUT#15,A,B$,C,D
30 PRINTA,B$,C,D
```

```
40 CLOSE15
```

This will display the error code (A), an error message (B$) and the track and sector of the error (C & D).

Now, in the case of the Block-Allocate command, you're really only interested in B$,C and D. If B$ is "OK" then the track and sector you selected have been allocated on the BAM. If B$ is not "OK" then C and D will contain the next available track and sector (Note: these will not have been allocated and you will have to perform the Block-Allocate instruction again to allocate the new track and sector).

Here's a routine that will look for, allocate and display the location of the next available track and sector on a diskette.

```
10 OPEN15,8,15
20 PRINT#15,"B-A:"0;1;1
30 INPUT#15,A,B$,T,S
40 IFB$="OK"THENT=1:S=1
   :GOTO60
50 PRINT#15,"B-A:"0;T;S
60 PRINT"TRACK=";T,
   "SECTOR=";S
70 CLOSE15
```

Storing Data

The track and sector information can be used to write to the disk also. Here's a data entry routine that puts data into the direct access channel:

```
100 OPEN2,8,4,"#"
110 I=I+1
120 PRINT"ITEM";I
130 INPUTX$
140 IFX$="XXX"THEN170
150 PRINT#2,X$
160 GOTO20
170 END
```

Now all you have to do is transfer the data from the direct-access channel to the diskette. To do this you can use the Block-Write command as follows:

```
170 PRINT#15,"B-W:"4;0;T;S
```

and CLOSE the channels:

```
180 CLOSE2:CLOSE15
```

System Capabilities

Although we've only covered four of the Disk Operating System commands, we now have enough information to write a complete mailing list system. The last article in this series will include a mailing list program and some guidelines to help you add as many features to it as you might need. **C**

VIC-20™

VIC-20 VIC-20 VIC-20 VIC-20 VIC-20

VIC-20 INTERFACING BLUE BOOK

Did you know that your VIC can be used to control a 99¢ toy motor so effectively that it runs like a precision machine? Or that you can build an accurate digital thermometer using the VIC and four parts costing less than $5?

These and other 18 interfacing projects selected for usefulness, ease of construction and low cost are detailed in the VIC-20 Interfacing Blue Book, a veritable gold mine of practical information on how to build a variety of interfaces for your computer.

Projects include: Connecting VIC to your stereo; Pickproof digital lock; Capacitance meter; Liquid level sensor; Telephone dialer; Voice output; 8K/16K RAM/ROM expansion; 128K RAM expansion; 8-bit precision D/A; 8-bit A/D converter; MX-80 interface and more.

Written by a college professor in a friendly and informative style, the Blue Book gives you theory of operation, schematics, program listings, parts list, construction hints and sources of materials for each one of the 20 projects.

If you want to get the most out of your VIC this book is a must. Cost is $14.95 (less than 75¢ per project!). Price includes postage.

VIC-20 VIC-20 VIC-20 VIC-20 VIC-20 VIC-20 VIC-20

VIC 20 is a trademark of Commodore Electronics Ltd

microsignal Dept C6
P.O. BOX 22
MILLWOOD NY 10546

Please send me a copy of the Blue Book.
Enclosed my check for $_____
NAME ____________________
ADDRESS ____________________
____
____

Above prices include postage in the U.S. CA res. add 6% tax. Foreign add $2.

---

software for less

VIC 20
COMMODORE 64

for education,
recreation
& business

MONOPOLY-64	19.95
APE CRAZE-64	24.95
PEARL DIVER-64	19.95
LOGGER-64	24.95
INTRUDER SCRAMBLER	17.95
CHOPLIFTER	34.95
EASY SCRIPT-64 Word Processor	
Reg. 99.95	84.95
DATA-MANAGER	
VIC-20	69.95
Commodore-64	89.95
GENERAL LEDGER	
VIC-20	89.95
Commodore-64	99.95

Commodore 64 & VIC is a reg. trademark of COMMODORE Bus. Machines, Inc.

Send check or money order. COD add $2.00. Shipping $1.50

pyramid (609) 386-9353
computerware

278 Warren St., Edgewater Park, NJ 08010

Tricky VIC Dynamic Bookkeeping

by Ron Kushnier

Type and save this bookkeeping program for the unexpanded VIC 20, with the option to expand it later.

My primary objective for developing a bookkeeping program was to provide a utility that was friendly, useful, and fast. To meet this objective, I wrote "PET Dynamic Bookkeeping", which appeared in the February, 1983, issue of *COMPUTE!* magazine.

The cassette-based program used dynamic keyboard techniques to automatically change system pointers, thus saving variable and array information within the program. This eliminated the need for DATA statements or slow data-tape files. The dynamic keyboard is a whole subject by itself. Suffice it to say that it is a technique that allows the computer to modify its own program by POKing into the keyboard buffer.

When I added the fast "Rabbit ROM" cassette utility to my system, I was able to load 10K worth of program and data in less than a minute. This was a time-frame I could live with.

To my surprise, most of the feedback from my article came from the VIC community rather than PET people. They wanted to know whether a similar program could be written for the VIC.

This turned out to be relatively straightforward. It was just a matter of adjusting a few pointers and reformatting some PRINT commands. The real challenge came when I tried to compress the program into an unexpanded VIC. The result is a much more refined program, which is set up for easy expansion when additional memory is added.

Sample Run

The program provides the following screens:

```
BOOKKEEPING PROGRAM

BILLING LIST
   1 CHARGES (81.31)
   2 GASOLINE (23.54)
   3 UTILITIES (204.05)
   4 END PROGRAM

WHAT WOULD YOU LIKE TO SEE? TYPE 1 TO 4
1

CHARGES (81.31)
   1 GIMBELS (68.53)
   2 PENNEYS (12.78)
   3 SEARS (0)
   4 BILLING LIST

WHAT WOULD YOU LIKE TO SEE? TYPE 1 TO 4
1

GIMBELS (68.53)
   1  JAN  ...22.56
   2  FEB  ...45.97
   3  MAR ...0
   4  APR  ...0
   5  MAY  ...0
   6  JUN  ...0
   7  JUL  ...0
   8  AUG  ...0
   9  SEP  ...0
   10 OCT  ...0
   11 NOV  ...0
   12 DEC  ...0
CHANGE THE LIST (Y/N)
Y
TYPE MONTH (1-12)
THEN HIT RETURN KEY
3
MAR...0

INPUT AMOUNT. THEN HIT RETURN KEY.
34.23

MAR...34.23

RECORD ALL DATA? (Y/N)
N

CHARGES (115.54)
   1 GIMBELS (102.76)
   2 PENNEYS (12.78)
   3 SEARS (0)
   4 BILLING LIST
```

```
WHAT WOULD YOU LIKE TO SEE? TYPE 1 TO 4
4

BILLING LIST
   1 CHARGES (115.54)
   2 GASOLINE (23.54)
   3 UTILITIES (204.05)
   4 END PROGRAM

WHAT WOULD YOU LIKE TO SEE? TYPE 1 TO 4
4

OK TO SAVE PROGRAM
```

Program Explanation

Line 0—This line jumps to the subroutine that adjusts pointers to include data. It is also used to jump over the DIM statement on line 5.
Line 5—Dimensions the needed arrays.
Lines 10-90—Initializes the program and defines the names used on the screens.
Lines 100-330—Displays the billing list and data lists to be updated. They also direct the user to the SAVE routine.
Lines 340-380—A subroutine used to bring back the number of a selected entry.
Lines 390-420—A subroutine used to select a YES or NO (Y/N) answer.
Lines 430-470—A subroutine used to return the number of a month.
Lines 480-590—This is a routine used to allow input without falling out of the program if just the RETURN key is hit.
Lines 600-640—This subroutine prints out the monthly amounts to the screen.
Line 650—A subroutine used to print the month and associated amount.
Lines 670-710—This routine is used to question whether a change in the list is desired.
Lines 720-770—This is a subroutine that is used to update the data.
Lines 58000-59090—This is the SAVE routine written by Howard Bicking. It adjusts pointers and creates program lines necessary to save variables and array data along with the program. Note that lines 58130 and 59010 through 59030 change dynamically within the program.

Initial Save

Once the program has been typed into the VIC and modified to individual needs, type RUN5. This will set up the arrays. After entering the data and going through the dynamic SAVE subroutine, save the program in the normal manner. Next time it is loaded, it may be run using a normal RUN command.

An Eye Toward Expansion

Although this program will run on an unexpanded VIC, the user may want to add more categories such as taxes, insurance, etc. Or, he may have more charge cards or additional utilities. To do this requires more memory.

The program is easily expanded to provide this capability. In the DIM statement on line 5, array "Q" is coded as:

Q(category, maximum number of entries, months)

In the listing shown, there are three categories; CHARGES, GASOLINE, and UTILITIES. There are also three entries under each category. Therefore we have Q(2,2,12). Note that the zero element of the array is used in the first two numbers. This is not so for the number of months, however.

Similarly, array "A" and "A$" are coded with the category number and the entry number.

Line 20 mirrors the first two numbers in the "Q" array.

The additional names of categories and entries are defined in a manner similar to those found in lines 30 through 60.

That is all there is to it unless there are over ten categories. It would then be necessary to dimension arrays "BL" and "C" as found in line 760.

Of course with additional memory comes additional waiting time involved for saving and loading the program. The day is saved again by Eastern House Software. They have a "Rabbit ROM" for the VIC to speed things up considerably. C

VIC Dynamic Bookkeeping

```
0 GOSUB59010:GOTO10
5 DIMQ(2,2,12),M$(12),A(2,2),A$(2,2)
10 PRINT"♥BOOKKEEPING PROGRAM":FORH=1TO700:NEXT
20 C=2:N=2
30 BL$(0)="CHARGES":BL$(1)="GASOLINE":BL$(2)="UTILITIES"
```

```
40 A$(0,0)="GIMBELS":A$(0,1)="PENNEYS":A$(0,2)="SEARS"
50 A$(1,0)="CASH":A$(1,1)="AMOCO":A$(1,2)="ARCO"
60 A$(2,0)="BELL":A$(2,1)="GAS":A$(2,2)="PECO"
70 M$(1)="JAN":M$(2)="FEB":M$(3)="MAR":M$(4)="APR":M$(5)="MAY":M$(6)="JUN"
80 M$(7)="JUL":M$(8)="AUG":M$(9)="SEP":M$(10)="OCT":M$(11)="NOV":M$(12)="DEC"
90 Z1$="RECORD ALL DATA?(Y/N)":Z$="CHANGE THE LIST?(Y/N)"
100 PRINT"♥BILLING LIST":PRINT"------------"
110 FORI=0TOC:PRINTI+1BL$(I);" (";BL(I);")":NEXT:PRINTC+2"END PROGRAM":T=C+2
120 GOSUB340:F=A-1:IF A=C+2THEN330
130 F=A-1:IF A=C+2THEN 330
140 GOSUB 660
150 FORI=0TON
160 IFA$(F,I)=""THEN180
170 PRINTI+1A$(F,I);"(";A(F,I);")":NEXT
180 PRINTI+1"BILLING LIST
190 T=I+1
200 GOSUB340:IFA=I+1THEN100
210 I=A-1:PRINT"♥"A$(F,I);"(";A(F,I);")":PRINT"---------":GOSUB600:GOSUB670
220 IF F1<>0THENF1=0:GOTO140
230 GOSUB480:Q(F,I,A)=V+Q(F,I,A)
240 A(F,I)=0:GOSUB720:A(F,I)=X
250 BL(F)=0:FORK=0TO T-2
260 BL(F)=BL(F)+A(F, K):NEXTK
270 GOSUB650
280 PRINT"▮"Z1$
290 GOSUB400
300 IFA$="N"THENGOTO140
310 GOTO58000
320 GOTO140
330 GOTO58000
340 PRINT"▮WHAT WOULD YOU LIKE TO SEE? INPUT 1 TO";T
350 GETA$:IFA$=""THEN350
360 A=VAL(A$)
370 IFA<1ORA>TTHEN350
380 RETURN
390 REM
400 GETA$:IFA$=""THEN400
410 IFA$<>"Y"THENIFA$<>"N"THEN400
420 RETURN
430 PRINT"▮TYPE MONTH(1-12)":PRINT"THEN HIT RETURN KEY
440 GOSUB490
450 A=V
460 IFA<1ORA>12THEN440
470 RETURN
480 PRINT"▮INPUT AMOUNT. THEN HIT RETURN KEY
```

```
490 AA$=""
500 OPEN1,0
510 GET#1,A$
520 IF A$<>CHR$(20)THEN550
530 IFLEN(AA$)-1 <0THEN510
540 AA$=LEFT$(AA$,(LEN(AA$)-1)):PRINT"▮ ▮";:GOTO510
550 PRINTA$;
560 AA$=AA$+A$
570 IFA$<>CHR$(13)THEN510
580 V=VAL(AA$)
590 CLOSE1:RETURN
600 FORJ=1TO9
610 PRINTJ;" "M$(J);"...";TAB(9);Q(F,I,J):NEXT
615 FORJ=10TO12
620 PRINTJ;M$(J);"...";TAB(9);Q(F,I,J):NEXT
640 RETURN
650 PRINTM$(A);"...";Q(F,I,A):RETURN
660 PRINT"⌐⌧";BL$(F);"(";BL(F);")":RETURN
670 PRINT"⌧"Z$
680 GOSUB400
690 IFA$="N"THEN F1=-1:RETURN
700 GOSUB430
710 GOSUB650:RETURN
720 X=0
730 FORJ=1TO12
740 X=X+Q(F,I,J):NEXTJ:RETURN
750 FORK=0TO T-2
760 BL(F)=BL(F)+C(K):NEXTK
770 RETURN
58000 REM-SAVE PROGRAM BY HOWARD BICKING
58010 PRINT"⌐⌧⌧":FORI=2TO6 STEP2
58020 N4=I*5:GOSUB58190:N5$=N4$
58030 N4=PEEK(43+I):GOSUB58190:N6$=N4$:N4=PEEK(44+I):GOSUB58190
58050 PRINT"59"+N5$" POKE"43+I","N6$":POKE"44+I","N4$:NEXT
58060 PRINT"58130 POKE45,"N6$":POKE46,"N4$":END
58080 PRINT"GOTO 58110
58090 POKE 198,5:FORI=0TO7:POKE631+I,13:NEXT:PRINT"⌧":END
58110 PRINT    :PRINT"⌐⌧⌧⌧⌧▮▮▮OK TO SAVE PROGRAM
58130 POKE45,017:POKE46,051:END
58190 N4$=RIGHT$("00"+RIGHT$(STR$(N4),LEN(STR$(N4))-1),3):RETURN
58200 GOSUB 59010:END
59010 POKE 42 ,244:POKE 43 ,041
59020 POKE 44 ,100:POKE 45 ,042
59030 POKE 46 ,017:POKE 47 ,051
59090 RETURN
```

Unclutter Your Desk!

Put your Commodore computer in a desk of its own.

Sure it will look great, and the desk is small enough to fit almost anywhere.

But the real benefits are your personal comfort and efficiency.

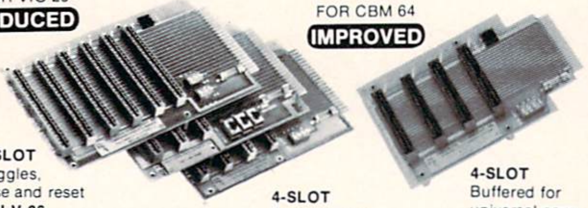
pearr DESK SYSTEMS

The Pearr Desk Systems
Box 323
Berrien Springs, MI 49103
616/473-3657

VIC 20™ and CBM 64™ EXPANDER BOARDS

FOR VIC 20 **REDUCED**

FOR CBM 64 **IMPROVED**



6-SLOT Toggles, fuse and reset **P/N V-36** **$69.95**
6-SLOT with 3-ft. ribbon cable **P/N V-46** **$89.95**

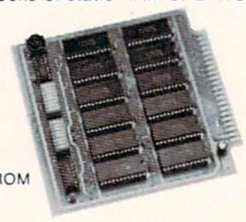
3-SLOT Slide switches and fuse **P/N V-23** **$49.95**

4-SLOT Toggles, fuse and reset **P/N V-24** **$59.95**

4-SLOT Buffered for universal compatibility. Toggles, fuse and reset. **$69.95**

NEW VIC 20™ MEMORY

A multipurpose board with sockets for three 8K blocks of static RAM or EPROM



Expands VIC 20 memory to 29K when equipped with 24K RAM

System reset button

Control switches for 8K memory blocks

6116 RAM or 2716 EPROM

90-day warranty

Gold connector

Sockets for all ICs

Sold with 8, 16 or 24K RAM

P/N VME-1
with 8K RAM **$79.95**
with 16K RAM **$99.95**
with 24K RAM **$119.95**

shown without cover

MONITOR/AUDIO CABLE

Connects VIC 20 or CBM 64 to audio amplifier and TV monitor

PRECISION TECHNOLOGY, INC.
COMPUTER PRODUCTS DIVISION
P.O. BOX 15454
SALT LAKE CITY, UTAH 84115
(801) 487-6266

Color 64 or VIC **P/N MC-2 $12.95**
B & W 64 only **P/N MC-3 $12.95**

See your dealer or place your order directly

VISA - M/C - CHECK - COD

TM-Trademark of Commodore Electronics Limited

Detecting Function Keys

on the Commodore 64

The function keys on the Commodore 64 are a set of four spare keys located down the right hand side of the keyboard. When using a basic program it is often desirable to display a message such as:

"Press F1 to continue"

The Commodore 64 makes it possible to detect these keys because each one has a CHR$ code assigned to it. These codes range from 133 to 140 for keys F1 through to F8. A simple way to wait for the F1 key to be pressed is as follows:

```
100 GET A$: IF A$<>CHR$(133) THEN 100
```

However, if you do not want to use CHR$ every time you want to check for a function key, then you can use the following method. If you type a single quote and then press each of the function keys in turn, you will see a series of reversed video graphics appear. We can use this property of the function keys to make the test loop a little easier to key in, as follows:

```
100  GET A$: IF A$<>"[F1]" THEN 100
```

Please note that the 'F1' in square brackets means press that key, not type it as shown; you should have a solid square with a line through the top of it between the quotes.

Another way of testing for the function keys (and any other keys for that matter) is to PEEK memory location 197 which contains a special code that tells the computer which row and column the key being pressed is on. This location contains 64 when no keys are depressed and a different number if a key is held down. We can use this facility to monitor the function keys all of the time and even wait for a key to be released before continuing with the rest of the program. The following codes are put into location 197 when each of the function keys are pressed:

F1 = 4
F2 = 5
F3 = 6
F4 = 3

For example:

```
 10 IF PEEK(197)<>4 THEN 10: REM
    WAIT FOR F1
 20 IF PEEK(197)<>64 THEN 20:REM
    WAIT FOR RELEASE
```

will wait for the F1 key to be pressed and then released before continuing with the rest of the program. C

—Courtesy of Commodore (U.K.)

User-Defined Function Keys

The BASIC program following will allow you to define the function keys so they print anything you like up to eight characters long. Lines 200-330 POKE the machine code into $C000 onwards. If the program stops at line 320 and prints the error message, then you have made a mistake in typing the data and should go through again and check it very carefully. If the data is POKEd in successfully then you will be able to use the function keys as if each one were a single key with a word on it. **C**

—Courtesy Commodore (U.K.)

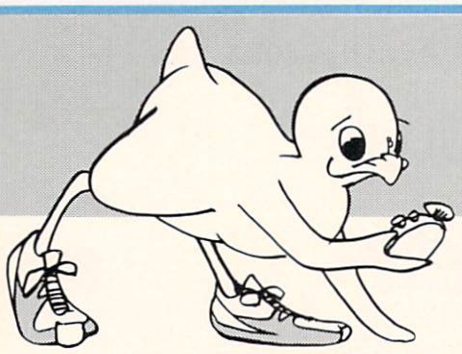
```
0 REM FUNCTION KEY DEFINE BY S BEATS
1 GOSUB 300                :REM POKE IN DATA
2 F$(1)="LIST"+CHR$(13)    :REM YOU CAN CHANGE THESE
3 F$(2)="RUN"+CHR$(13)     :REM DEFINITIONS IF YOU WANT
4 F$(3)="GOSUB"            :REM BUT REMEMBER IT IS 8 CHARACTERS
5 F$(4)="SAVE"+CHR$(34)    :REM MAXIMUM PER KEY
6 F$(5)="LOAD"
7 F$(6)="GOTO"
8 F$(7)="CHR$("
9 F$(8)="RETURN"
10 SYS12*4096+64           :REM ENABLE CODE
20 V=12*4096-1             :REM START OF DATA
30 FORI=1TO8:K=I-1:V1=V+K*8
40 FORJ=1TOLEN(F$(I))      :REM POKE IN KEY DEFINITIONS
50 POKE V1+J,ASC(MID$(F$(I),J,1))
60 NEXT:NEXT:END
200 DATA 120,169,87,141,20,3,169,192,141,21
210 DATA 3,88,162,63,169,0,157,0,192,202
220 DATA 16,250,96,165,197,201,64,208,6,141
230 DATA 151,192,76,148,192,205,151,192,240,44
240 DATA 141,151,192,162,3,221,152,192,240,5
250 DATA 202,16,248,48,29,138,174,141,2,240
260 DATA 3,24,105,4,10,10,10,168,162,0
270 DATA 185,0,192,157,119,2,200,232,224,8
280 DATA 208,244,134,198,76,49,234,64,4, 5,6,3,0
300 FORI=0TO92:READA:Z=Z+A
310 POKE12*4096+64+I,A:NEXT
320 IF Z<>10771 THENPRINT"ERROR IN DATA STATEMENTS":STOP
330 RETURN
```

Commodore 64 Screen Dump

(Updated from the 1525 Printer Manual)

This program will allow you to dump the Commodore 64 screen to your 1525 printer. **C**

```
  1 GOSUB100
  2 END
100 SI$=CHR$(15):BS$=CHR$(8):PO$=CHR
    $(16)
110 RV$=CHR$(18):RO$=CHR$(146):QT$=
    CHR$(34)
120 MF$=CHR$(145):VR=PEEK(648)*256
130 OPEN4,4:PRINT#4
140 FORCL=0TO24:QF=0:AS$=MF$:FORRO=
    0TO39
150 SC=PEEK(VR+40*CL+RO)
160 IFSC=34THENQF=1-QF
170 IFSC<>162THEN200
180 QF=1-QF:IFQF=1THENAS$=AS$+RV$+QT$
    :GOTO260
190 AS$=AS$+QT$+RO$:GOTO220
200 IFQF=1AND(SC>128)THENSC=SC-128:
    GOTO220
210 IFSC>=128THENSC=SC-128:RF=1:AS$=
    AS$+RV$
220 IFSC<32ORSC>95THENAS=SC+64:GOTO250
230 IFSC>31ANDSC<64THENAS=SC:GOTO250
240 IFSC>63ANDSC<96THENAS=SC+32:
    GOTO250
250 AS$=AS$+CHR$(AS)
260 IFRF=1THENAS$=AS$+RV$:RF=0
270 NEXTRO
280 IFQF=0THENPRINT#4,SI$PO$"20"AS$:
    GOTO300
290 PRINT#4,SI$+PO$+"20"+AS$+QT$
300 NEXTCL:PRINT#4,SI$:CLOSE4:RETURN
```



It's Time for TOTL SOFTWARE!

for the VIC 20™ and COMMODORE 64™
WORD PROCESSING AND MAILING LIST & LABEL
now available with **CHICKSPEED**

FAST PRINTING • LIGHTNING LOADS • SIMPLE COMMANDS

TOTL.TEXT 2.0 + CS VIC + 8K expansion	$25.00
TOTL.TEXT 2.5 + CS VIC + 16K expansion	$35.00
TOTL.TEXT 2.6 + CS Commodore 64	$40.00
TOTL.LABEL 2.1 + CS VIC + 16K expansion	$20.00
TOTL.LABEL 2.6 + CS Commodore 64	$20.00
TOTL TIME MANAGER 2.1 VIC + 8K expansion	$30.00
TOTL TIME MANAGER 2.6 Commodore 64	$35.00
time management, scheduling, reports	
RESEARCH ASSISTANT 2.0 VIC + 8K expansion	$30.00
RESEARCH ASSISTANT 2.0 Commodore 64	$35.00
key word cross-reference research tool	
TOTL.BUSINESS 3.0 VIC + 16K expansion	$85.00
TOTL.BUSINESS 3.6 Commodore 64	$95.00
business programs require disk and are shipped on disk	
One Megabyte Fuzzy Diskette	$25.00
computer novelty pillow	

All programs work with 40/80 column (VIC) and 80 column (64) adapters—compatible with tape or disk systems —shipped on cassette tape— available on disk $4.00 extra.

Quality You Can Afford
Available at your local dealer or by phone order

TOTL
software inc.

1555 Third Ave., Walnut Creek, CA 94596
Call (415) 943-7877

Commodore 64 and VIC 20 are registered trademarks of Commodore Electronics, Ltd.

Program, Save Yourself

by Bruce Jaeger

A nifty utility for the Commodore 64

Common sense tells us that we should make copies of our programs as we work on them, to prevent losing hours of work due to an accidental "crash" or the kids pulling the plug to make Daddy laugh. (He doesn't.) But it is a bit tiresome to keep scratching the old program on the disk and SAVEing the new. Especially when the computer will do it for you!

Load UTILITYPRG before you begin programming, and leave it attached to your program until you're finished and don't need it anymore. When called by either RUN 63000 or GOTO 63000, the program will give you the time of day, and the choice of SAVEing the program, or LISTing all or part of it, saving you all the bother of opening and closing files.

The only "customizing" you need do is change PG$ in line 63000 to the name of your program. **C**

```
CONTROL CODE KEY

"⊐" = CLEAR
"⌂" = HOME CURSOR
"⌉" = CURSOR UP
"▣" = CURSOR DOWN
"▌" = CURSOR LEFT
"▐" = CURSOR RIGHT
"▩" = REVERSE
"■" = REVERSE OFF
```

```
63000 PG$="UTILITYPRG":E$=CHR$(1):PRINT"⊐"
63010 PRINT"⌂                    ▩";PG$
63020 GOSUB63290:PRINT"⌂";H$;":";M$;":";RIGHT$(TI$,2)
63030 PRINT"▣▣▩S■CRATCH OLD PROGRAM, SAVE NEW."
63040 PRINT"▣▣▩L■IST PROGRAM ON CBM PRINTER"
63050 PRINT"▣▣▩T■IME OF DAY CHANGE"
63060 PRINT"▣▣▩R■UN PROGRAM"
63070 PRINT"▣▣▣▩SELECT OPTION■"
63080 GETS$:IFS$=""THEN63020
63090 IFS$="T"THEN63150
63100 IFS$<>"R"AND S$<>"T"ANDS$<>"S"ANDS$<>"L"THENPRINT"⊐":END
63110 IFS$="L"THENINPUT"▣▣TODAY'S DATE   00/00/83▌▌▌▌▌▌▌▌▌▌▌";TD$:GOTO63180
63120 IFS$="R"THENRUN
63130 IFS$="S"THENINPUT"▣▣PROGRAM ON DRIVE  0▌▌▌";DR$:DR$=DR$+":":GOTO63170
63140 PRINT"⊐":END
63150 PRINT"▣SET THE TIME:  HRMNSC"
63160 INPUT"                   000000▌▌▌▌▌▌▌▌▌";TI$:GOTO63000
63170 CLOSE1:OPEN1,8,15:PRINT#1,"S"+DR$+PG$:SAVEDR$+PG$,8:GOTO63000
63180 OPEN4,4:PRINT#4,E$;E$;PG$:PRINT#4
63190 PRINT#4,E$;"VERSION AS OF ";TD$;"  ";:GOSUB63290:PRINT#4,H$;":";M$
63200 INPUT"▣WANT PAGING   N▌▌▌";WP$:IFWP$<>"N"THENPRINT#4,CHR$(147);
63210 INPUT"▣WANT ▩A■LL OF THE PROGRAM, OR ▩P■ART   A▌▌▌";AP$
63220 IFAP$="A"THENL=0:H=62999:GOTO63250
63230 INPUT"▣FROM LINE #   0▌▌▌";L
63240 INPUT"▣  TO LINE #   62999▌▌▌▌▌▌▌▌";H
63250 PRINT"⊐PRESS ▩RETURN■ TO LIST"
63260 PRINT"⌂▣▣▣▣▣CMD4:LIST";L;"-";H
63270 PRINT"⌂▣▣▣▣▣▣▣PRINT#4:CLOSE4:GOTO63000"
63280 PRINT"⌂▣▣":END
63290 H$=LEFT$(TI$,2):M$=MID$(TI$,3,2):H$=STR$(VAL(H$)):RETURN
```

Machine Language Monitor in the Upgrade PET

by Elizabeth Deal

The machine language monitor (MLM) in the PET is, to me, one of the most valuable hunks of code. Often things are easier to do using the monitor instead of BASIC commands. For instance, it is simple to type ".M 0028 0029" to see where BASIC begins. Typing PRINT PEEK(40), PEEK(41) or, worse, PEEK (40)+256*PEEK(41), as you must in BASIC, is a pain. There are many other monitor commands that also help simplify life, and the monitor provides security in the event BASIC has been damaged. But there are small troubles in the upgrade version of the PET. However, if we know how to handle them we shouldn't have any problem.

The LOAD (.L) Command

Problem 1: In the February, 1983, issue of this magazine I asked if anyone knows why only one byte of a file loads into the upgrade PET when a machine language monitor ".L" command is used.

I do. Here is the answer: It doesn't always happen. It happens when ST is set to a non-zero value and is not cleared.

Incidentally, a very reliable source tells me that the Commodore 64 version of the Supermon, which includes all the monitor commands, does not suffer from these problems. It was written by Jim Butterfield and is in *COMPUTE!* issue 32, January, 1983.

A bit more detail: There is a lengthy routine in ROM, at $F3C2, for LOADing programs into the PET via BASIC. The system vector is at $FFD5. One of the first things this routine does is examine and use the file name, device number, and so on. It then clears the ST byte (file status word) to zero. This code is at $F43E. A separate subroutine ($F322) is called to move the program bytes from disk into memory. It puts them in, verifies, and counts. On exit, the start of the variables pointer is changed to reflect the new program end, a CLR is performed, and you're ready to go.

The .L command in the MLM sets the file parameters and then uses only the loading subroutine at $F322. It does not change BASIC pointers, of course. And it unfortunately fails to clear ST prior to loading. Hence, if you came in with ST=64, it so remains. PET thinks the job is done after one byte, and blinks the cursor at you. The illusion of a completed load can be disastrous when you try to use the code, since the code isn't there.

Several procedures leave ST=64 on. One of those, unfortunately, is the input of the error status in the disk (a wedge command, > "greater than", or the whole INPUT#15,E1, etc., version). At the completion of the status report, ST is set to 64. Another procedure that leaves ST=64 on is a successful .L load. But if you enter the monitor with no intervening ST clearing commands, you'll get zapped.

Several procedures leave ST=64 on. One of those, unfortunately, is the input of the error status in the disk (a wedge command, >, or the whole INPUT#15,E1, etc., version). At the completion of the status report, ST is set to 64. Another procedure that leaves ST=64 on is a successful .L load. But if you enter the monitor with no intervening ST clearing commands, you'll get zapped.

The cure is to clear ST yourself. Prior to entering the machine language monitor it can be cleared by POKE150,0. You can then enter the MLM and .L-load. From within the MLM setting $0096 to 00 does the trick, of course. And LOADing files from within a running BASIC program also is not disturbed by a bad ST, as that routine goes through the whole necessary procedure.

Problem 2: If a file to be loaded is not on the floppy a STOP key has to be used. That's fine in BASIC (most of the time), but could be bad in the MLM since that particular STOP-test routine exits to BASIC. Should BASIC be clobbered so a file is not on disk, use of the STOP key dispatches you right out of the safe monitor into the hostile world of nonexistent BASIC. Ouch! So, if you think BASIC is in trouble, be careful what you try to load.

The Executive Code (.G) Command

If you have ever wondered why after saving or loading (.S or .L) a file you crash on a subsequent .G command, the reason is that the file name and the temporary storage for the IRQ vector ($0207) share memory. So you do a .G, and when the job is done, the PET tries to stuff two bytes of your file name into its IRQ vector (normal housekeeping), with rather unpleasant results. You may uncrash, of course, using the Butterfield procedure, but if the silly new IRQ code damaged BASIC or the machine language program, you're in serious trouble.

The remedy is very simple: ask for register display and change the value under IRQ to E62E or whatever you think should be there. You can then use the .G command safely.

If you wish to snoop, the monitor's .G command is at

$FECF; the SAVE and LOAD commands are at $FF11.

Three Big Solutions

1. Micromon solves some problems and it includes DOS-wedge commands. But Micromon is big, not easily relocatable, sometimes conflicts with IRQ, and uses a lot of memory in the first tape buffer.

2. The POWER chip, available from Professional Software, solves the .L and .G problems by permitting you to LOAD programs from BASIC. You use FIX or POWAID's PTRFIX to take care of BASIC pointers. When you have POWER there is little reason to *ever* use the .L command anyway.

3. BASIC 4 apparently does not suffer from any of these troubles; if you can afford to obsolete some of your favorite chips and programs (not me!), switch over.

I thank Jim Butterfield for a bit of guidance in answering some of those questions. C

New Commodore 64™ Games!

An introductory offer from

ADVANCED ARCADEWARE

HEAD ON
A daring and dangerous death race.

ATOMIC MAN
Super hero rescues a city.

ALIEN INVASION
4 players. Creatures descend.

SERPENTS
2 players. Snakes entangle each other.

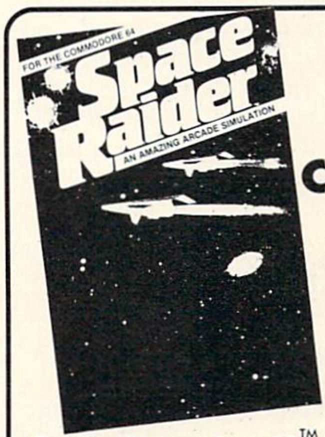
4 game pack suggested list price $59.95

NOW $29.99

To order: Mail address along with check or money order to:

ADVANCED ARCADEWARE
P.O. Box 845
Thomasville, NC 27360
(919) 431-3231

Specify Disk-Pack or Cassette Pack
Add $2.00 for shipping Allow 2 weeks for delivery



An Arcade Simulation for the Commodore 64™

Your mission is to destroy the enemy ships

$19.95

For the Commodore 64:™

ABC Fun Part of a series of educational programs for preschool aged children. $9.95

File Organizer Creates Data Base storage system. Useful for mailing lists, club rosters, check books, etc. $29.95

Sprite Editor The easy way to create, copy, alter, and save up to 224 sprite shapes. $24.95

Spread Sheet Powerful calculation and report formatting capabilities. Memory divided into grid: 26 rows by 10 columns. $34.95

Dissassembler Familiar with 6502 assembly code? This will decode ROM or help debug RAM. $19.95

Super Roller Challenging dice game with sprite graphics and sound. Yahtzee-style rules of play. $14.95

Cross Reference Generator for BASIC programs Displays line numbers in which any word of BASIC vocabulary appears. Allows you to change variable name and ask for lines where it appears, and more. $19.95

For the VIC-20:™

Caves of Windsor A cave adventure game. The object is to restore wealth and happiness to the small village of Windsor. $14.95

Burger & Fries Fast action joystick game. Eat the burgers and fries but avoid the shakes for a top score. $14.95

Catalog available
Dealer inquiries invited
PHONE ORDERS: (703) 491-6502
HOURS: 10 a.m. to 4 p.m. Mon.—Sat.

Virginia Micro Systems
13646 Jeff Davis Hwy
Woodbridge, VA 22191

Commodore 64 and VIC-20 are registered trademarks of Commodore Electronics Ltd.

The Beep Wedge for the PET

by Elizabeth Deal

Computers you see on TV make all sorts of blip-beep-blap noises as they process very important information. Real computers don't seem to do this sort of thing. To remedy the situation here is a noise-maker for PET or CBM computers, all versions except original ROMs. The beep idea comes from Keith Falkner's tutorial article about the Apple computer in issue 32 of *COMPUTE!*

As each and every character in a program or a direct command is interpreted by the PET, my routine causes the system to detour to a CB2-sound routine prior to acting on a character. The process is identical to that of the well known DOS-wedge, Toolkit, and POWER. However, for the sake of brevity and education a primitive manual hookup method is shown here.

The pitch of the sound is inversely proportional to the character or keyword token the PET sees. Some low values are inaudible or produce a high squeaky pitch, but the short duration of the beeps prevents you from going crazy.

The detour takes a bit of time; PET will process everything in sight, but very slowly. Once enabled, the routine can be tried during a program run. (Pick a program with few loops, as they are boring.) It can also be tried on direct commands, such as:

LIST
LIST1-63999
PRINT"SILENCE. CHRGET DOES NOT LOOK HERE"
PRINT some very long expression; the more things to do the merrier it gets.

To begin this foolishness you have to hook it up. Enter the machine language monitor using SYS4 and follow the listing in lines 105-117. Don't bother saving if your typing is good. As you ask for a display of locations $0070+ write down the contents of the first three bytes. We'll change them for making sound. To fix the PET you'll need to reset or put them back in later. I told you it was a primitive hookup!

You do not really have to put this code at $4000 (16384 decimal). It is relocatable and can be placed anyplace you like, but you mustn't forget to tell locations $0071-72 where the code is (rightmost two digits of the address go first, then the leftmost digits into $72). Just follow the pattern you see on the screen. Needless to say, unless protected by setting the top-of-memory pointer (POKE a zero into 48 and 52, POKE 64 into 49 and 53), this code will vanish on long LOADs or large arrays, and you'll have a mess. But for minor testing, being in the middle of a 32K PET it's rather immune from disaster.

You can change the timing of the notes by changing the hold time at location $401C. Replacing $10 with $50 sounds good; $04 there is OK for working the disk, which is usually in a boring loop. Do you own thing.

What we have just done is a tiny little building block for adding commands to BASIC or tracing a running program. We got a character and used its value to beep at a certain pitch; that's all. Had this been a DOS-wedge program, the character might have been an up-arrow, in which case a routine to LOAD a program would have been invoked, for instance.

Once again: what is missing is an elegant hookup. Start and kill-type commands are just not there. Coexistence with other utilities is not there. These features are harder to write, but are, of course, essential. For more on this subject snoop in the DOS-Wedge or your favorite utility via the SUPERMON. **C**

```
1 REM"S=SA"@1:SNDTR.LIST",8
2 :---------------------------------
3 :SOUND TRACE     ELIZABETH DEAL
4 :---------------------------------
100 :SYS4
101 :
102 :B*
103 :      PC  IRQ  SR AC XR YR SP
104 :.;   0005 9053 30 00 5E 04 F6
105 :.M 0070 0070
```

```
106 :.:  0070 4C 00 40 02 E6 78 AD 02
107 :.M 4000 402F
108 :.:  4000 8A 48 98 48 A0 00 E6 77
109 :.:  4008 D0 02 E6 78 A9 10 8D 4B
110 :.:  4010 E8 A9 0F 8D 4A E8 B1 77
111 :.:  4018 8D 48 E8 A2 10 A0 00 88
112 :.:  4020 D0 FD CA D0 F8 8E 4B E8
113 :.:  4028 68 A8 68 AA 4C 76 00 AA
114 :.S"0:SOUNDTRACE",08,4000,402F (DISK)
115 :       ---OR---
116 :.S"SOUNDTRACE",01,4000,402F (TAPE)
117 :.X
118 :READY.
119 :-----------------------------------
300 :REM DISSASSEMBLY
301 :4000 8A          TXA          ;HOUSEKEEPING
302 :4001 48          PHA
303 :4002 98          TYA
304 :4003 48          PHA
305 :4004 A0 00       LDY #$00
306 :4006 E6 77       INC $77      ;CHRGET PET CODE
307 :4008 D0 02       BNE $400C
308 :400A E6 78       INC $78
309 :400C A9 10       LDA #$10     ;CB2 SOUND
310 :400E 8D 4B E8    STA $E84B
311 :4011 A9 0F       LDA #$0F
312 :4013 8D 4A E8    STA $E84A
313 :4016 B1 77       LDA ($77),Y
314 :4018 8D 48 E8    STA $E848
315 :401B A2 10       LDX #$10     ;HOLD IT
316 :401D A0 00       LDY #$00
317 :401F 88          DEY
318 :4020 D0 FD       BNE $401F
319 :4022 CA          DEX
320 :4023 D0 F8       BNE $401D
321 :4025 8E 48 EB    STX $E84B    ;STOP IT
322 :4028 68          PLA          ;HOUSEKEEPING
323 :4029 A8          TAY
324 :402A 68          PLA
325 :402B AA          TAX
326 :402C 4C 76 00    JMP $0076    ;GIVE PET A CHANCE
327 :-----------------------------------
```

Where Are We?

by Elizabeth Deal

It is often desirable to know where BASIC or machine code is executing and this article will cover several ways of getting that information. It was developed on an Upgrade PET, should work on all PETs and might be transferable to other 6502 systems.

Why?

There are several reasons that we might wish to know where we are. The most obvious reason is for self-modifying programs. Usually this type of program POKEs data into a second or last line of BASIC, since those can easily be found. The method shown here will allow us to do that horrible activity anyplace. Another reason might be to reset a READ pointer to a particular group of items. Still another, simply to be able to look at the code via the monitor display.

In machine code I often find that I have an almost relocate-able program were it not for several bytes of data or one subroutine. In such cases a BASIC or machine code relocator is of value, but it often ends up being much longer than the tiny code I write, so the whole effort becomes silly.

There are many other reasons; these are just examples of what we might need.

BASIC Programs

Two pointers are quite useful. The "get the next BASIC character" pointer in the CHRGET routine ($77/78, dec 119/120) is live at all times and can be successfully used in machine code inserts into a BASIC program. For testing purposes, if we say SYS634, and in the first buffer at $027A we place a zero (BRK) then we'll see that this pointer points to the next character after the SYS call, namely a comma (in this case; it might well be a colon or an end-of-line zero). Subsequent JSR CHRGET places this character in the A-register and we can go on from there. People using BASIC 4 should be aware that BRK causes this pointer to be reset back to the start of BASIC text. Therefore they will have to arrange to save it before doing a break. (POWER users are familiar with this trick; it is used in the "SYS TRACE,parameters" command.)

The CHRGET pointer can answer the question "where are we?" very precisely. In my opinion, it should *not* be used in BASIC-only programs. The very action of interrogation modifies the pointer: if the inquiry crosses a page boundary, the returned value will be off by almost a page.

Another good pointer to know about is the pointer to the next statement to be executed after CONTinue. As the name implies, if the program is STOPped this pointer will contain a valid re-entry address. The address will be that of the next colon or end-of-line zero, hereafter referred to as "zero".

When STOP is not issued, the value of this pointer is invariant during statement execution. It points to the preceding terminator (colon or zero) until the next terminator is encountered.

So how do we find out where we are? Keeping in mind that the pointer does not change until a terminator and that it points to the beginning of the currently executing statement (zero or colon), this line will tell the location of the zero preceding line 250:

```
250 x=peek(58)+256*peek(59)
```

and this line will tell us the location of the colon preceding the x=peek... statement:

```
250 p=2:x=peek(58)+256*peek(59)
```

A function may be used since this pointer does not track the function definition, hence:

```
100 deffnpp(q)=peek(q)+256*
   peek(q+1)
•
•
•
250 x=fnpp(58)
```

returns a valid address.

The only restriction in this method is that the inquiry needs to be packed into *one* command so as not to change the statement pointer and run the risk of a page crossing mishap; lines such as

```
250 x=peek(58) :y=peek(59) or
250 pokez1,peek(58) :pokez2,peek(59)
```

return an absurd address if the high byte of the address changes on a line, which is when the colon has crossed a page. The bad address is *exactly* one page too high.

Now that we know how to pick up the location of the current line we can go one step further and find a location of the next line, which opens the door to all sorts of fun.

Having defined a function in line 100, we can now code

```
250 x=fnpp(58) :y=fnpp(x+1)−1
260 rem next line
```

X gives us the address of a zero preceding line 250. X+1 and X+2 give us the pointer to line 260. X+3 and X+4 give us the line number of 250 coded as low and high bytes. X+5 is the first position of BASIC text on line 250.

Y delivers the address of the end of

line 250, which is the same as a zero preceding line 260. Y+1 points to the chain for a line beyond 260, Y+5 is the address of the first character on line 260, a token for REM. Y+3 and Y+4 deliver the line number, 260.

Now we can do something useful. If we modify the pointer to a DATA item, held in $3E/3F (dec 62/63) with a value of X we can access the DATA list in lines equal to or greater than 250. If Y is used, we'll access DATA items in lines 260 and higher. We will use three common functions, an overkill in this situation, but they do belong in a program such as this. The routine at the end of this article shows how to do it.

When run, Q$ will receive "HERE" rather than "NOT HERE". Normally, had RESTORE been in force by virtue of RUN or CLR, "NOT HERE" would result. As you can see, the DATA pointer need not be set to a DATA line. It should be set to a terminator: a colon, zero or a comma in a DATA list. PET finds subsequent DATA items without our intervention.

Being able to position the DATA pointer permits us to successfully use subroutines containing their own data without having to worry about DATA lines physically preceding the ones we want. We do not need to issue numerous READ statements to bypass the unwanted list.

The same procedure can be used in self-modifying programs. We are no longer limited to destruction of good code from line 2 on, we can do it from any place we wish. But for the sake of preserving the code and our sanity we now have all the tools needed to test if the POKing destination exists and whether it contains sufficient room.

Small Machine Code Programs

If the program is entered by USR(v) then, of course, the address of the routine is known. It is held in locations 1 and 2.

If the program is entered via a SYS address then A contains the high byte and Y contains the low byte of the address. Also locations $11/12 (dec 17/18) hold the call address until the next BASIC instruction, including calls to many BASIC ROM routines, clobbers it.

If the program is not entered from BASIC then the stack can help us. Frank Covitz, of instrument synthesis fame, recommends doing a phony subroutine call. Go HUNTing for $60, he says. (Also, see Frank's article on bit-mapped graphics in this issue.)

The Supermon's HUNT command found a first in-ROM RTS instruction ($60) in my upgrade PET at $C207. There is no shortage of RTS in ROM, any old $60 will do. Now the trick is to go there by JSR xxxx and on return examine the stack two bytes back. As you can see, the A and Y registers hold the address−1. That's all there is to it.

We can use it as an ultimate method in writing relocate-able code. The point is that once we know where we are, we can have the PET modify the absolute addresses by adding a known value which we obtained from the stack.

Bear in mind that the method is good for small routines with only a few two-byte addresses internal to the program. The method is too tedious and practically useless in larger code.

Anyone using the Instrument Synthesis Software Package from Micro Technology Unlimited is probably aware of the fact that the program honors user-written subroutines. Since the placement of such code depends on your configuration string, the song data and the command string, knowing where you are is a good thing, because the three parameters can vary. And if we plan for such code to travel to other systems, we must observe MTU's policy of "must run on all 6502-s". Writing completely relocate-able code is the only sensible way, PEEKing the stack helps in that process.

References

1. Butterfield, Memory Maps
2. Butterfield, "Scanning the Stack", *COMPUTE!* #8 (Jan 81)

C

```
500 REM------------------------------
510 REM WHERE ARE WE   ELIZABETH DEAL
520 REM------------------------------
530 A1=58:A2=62
540 DEFFNPP(Q)=PEEK(Q)+256*PEEK(Q+1)
550 DEFFNHI(Q)=INT(Q/256)
560 DEFFNLO(Q)=Q-256*FNHI(Q)
570 DATA NOT HERE
580 Y=FNPP(FNPP(A1)+1)-1:POKEA2,FNLO(Y):POKEA2+1,FNHI(Y)
590 REM>READ POINTER HAS NOW BEEN SET TO THIS LINE
600 READQ$:PRINTQ$
```

```
610 DATA HERE
620 POKEY+6,94-PEEK(Y+6):REM NOW
     LIST
630 REM-------------------------
640 :
650 DISASSEMBLY - $C2D7 CONTAINS
     RTS
660 ($60) IN UPGRADE PET
670 :
1100 .,   4080 20 D7 C2    JSR
     $C2D7
1101 .,   4083 BA          TSX
1102 .,   4084 BD FF 00    LDA
     $00FF,X
1103 .,   4087 A8             TAY
1104 .,   4088 BD 00 01       LDA
     $0100,X
1105 .,   408B 00             BRK
1106 .R
1107 :     PC   IRQ  SR AC XR YR SP
1108 .;   4080 9053 30 00 5E 04 F8
1109 .G
1110 B*
1111 :     PC   IRQ  SR AC XR YR SP
1112 .;   408C 9053 30 40 F8 82 F8
1113 .X                   --    --
1114 READY.
1115 REM-------------------------
```

COMMODORE USERS

Join the largest, active Commodore users group. Benefit from:

— Access to hundreds of public domain programs on tape and disk for your **Commodore 64™, VIC 20™** and **PET®/CBM™**.

— Informative monthly club magazine **THE TORPET.**

Send $1.00 for Program & Information Catalogue.
(Free with membership).

Membership Fees for 12 Months		
	Canada	— $20 Can.
	U.S.A.	— $20 U.S.
	Overseas	— $30 U.S.

Toronto Pet Users Group Inc.
Department "C"
1912A Avenue Road, Suite 1
Toronto, Ontario, Canada M5M 4A1

* LET US KNOW WHICH MACHINE YOU USE *

VIC 20, Commodore 64, CBM are trademarks of Commodore Electronics Ltd. **PET** is a registered trademark of Commodore Business Machines, Inc.

New VIC 20, CBM and PET Products

COMMUNICATION BOARD
All you do is turn on your VIC and it's ready to connect to a modem. Makes the VIC into a dumb terminal.
Model MW-300 $39.95

V-DATA BASE CASSETTE
Allows user to input mail lists, and mini inventories, store on disk or cassette, and recall and print.
Model MW-301 $29.95

VIC 20/CBM 64 PRINTER INTERFACE
This unit will interface your VIC 20 or CBM 64 to standard parallel printers such as Epson, Centronics, Anadex, Paper Tigers, Okidata, and many others.

Model MW-302 $119.95

DIGITAL TO ANALOG CONVERTER OR 8 CHANNEL ANALOG TO DIGITAL CONVERTER
Allows your VIC or CBM to output analog signal or digitize up to 8 analog channels.
Model MW-304p PET $139.95
Model MW-304v VIC $129.95

Micro World Electronix Inc.
6340 W. Mississippi Ave.
Lakewood, Colorado 80226
(303) 934-1973

user groups

User Group Listing

ALABAMA

Huntsville PET Users Club
9002 Berclair Road
Huntsville, AL 35802
Contact: Hal Carey
Meetings: every 2nd Thursday

ALASKA

COMPOOH-T
c/o Box 118
Old Harbor, AK 99643
(907) 286-2213

ARIZONA

VIC Users Group
2612 E. Covina
Mesa, AZ 85203
Contact: Paul Muffuletto

Catalina Commodore Computer Club
2012 Avenida Guillermo
Tucson, AZ 85710
(602) 296-6766
George Pope
1st Tues. 7:30 p.m.
Metro Computer Store

Central Arizona PET People
842 W. Calle del Norte
Chandler, AZ 85224
(602) 899-3622
Roy Schahrer

ACUG
c/o Home Computer Service
2028 W. Camelback Rd.
Phoenix, AZ 85015
(602) 249-1186
Dan Deacon
First Wed. of month

West Mesa VIC
2351 S. Standage
Mesa, AZ 85202
Kenneth S. Epstein

ARKANSAS

Commodore/PET Users Club
Conway Middle School
Davis Street
Conway, AR 72032
Contact: Geneva Bowlin

Booneville 64 Club
c/o A. R. Hederich
Elementary School
401 W. 5th St.
Booneville, AR 72927
Mary Taff

CALIFORNIA

SCPUG Southern California PET Users Group
c/o Data Equipment Supply Corp.
8315 Firestone Blvd.
Downey, CA 90241
(213) 923-9361
Meetings: First Tuesday of each month

California VIC Users Group
c/o Data Equipment Supply Corp.
8315 Firestone Blvd.
Downey, CA 90241
(213) 923-9361
Meetings: Second Tues. of each month

Commodore Users Club
1041 Foxenwoods Drive
Santa Maria, CA 93455
(805) 937-4106
Contact: Greg Johnson

Valley Computer Club
2006 Magnolia Blvd.
Burbank, CA
(213) 849-4094
1st Wed. 6 p.m.

Valley Computer Club
1913 Booth Road
Ceres, CA 95307

PUG of Silicon Valley
22355 Rancho Ventura Road
Cupertino, CA 95014

Lincoln Computer Club
750 E. Yosemite
Manteca, CA 95336
John Fung, Advisor

PET on the Air
525 Crestlake Drive
San Francisco, CA 94132
Max J. Babin, Secretary

PALS (Pets Around)
Livermore Society
886 South K
Livermore, CA 94550
(415) 449-1084
Every third Wednesday
7:30 p.m.
Contact: J. Johnson

SPHINX
7615 Leviston Ave.
El Cerrito, CA 94530
(415) 527-9286
Bill MacCracken

San Diego PUG
c/o D. Costarakis
3562 Union Street
(714) 235-7626
7 a.m.-4 p.m.

Walnut Creek PET Users Club
1815 Ygnacio Valley Road
Walnut Creek, CA 94596

Jurupa Wizards
4526 Kingsbury Pl.
Riverside, CA 92503
Contact: Walter J. Scott

The Commodore Connection
2301 Mission St.
Santa Cruz, CA 95060
(408) 425-8054
Bud Massey

San Fernando Valley Commodore Users Group
21208 Nashville
Chatsworth, CA 91311
(213) 709-4736
Tom Lynch
2nd Wed. 7:30

VACUUM
277 E. 10th Ave.
Chico, CA 95926
(916) 891-8085
Mike Casella
2nd Monday of month

VIC 20 Users Group
2791 McBride Ln. #121
Santa Rosa, CA
(707) 575-9836
Tyson Verse

South Bay Commodore Users Group
1402 W. 218th St.
Torrance, CA 90501
Contact: Earl Evans

Slo VIC 20/64 Computer Club
1766 9th St.
Los Osos, CA

The Diamond Bar R.O.P. Users Club
c/o Rincom School
2800 Hollingworth
West Covina, CA 91792
(213) 965-1696
Don McIntosh

Commodore Interest Association
c/o Computer Data
14660 La Paz Dr.
Victorville, CA 92392
Mark Finley

Fairfield VIC 20 Club
1336 McKinley St.
Fairfield, CA 94533
(707) 427-0143
Al Brewer
1st & 3rd Tues. at 7 p.m.

Computer Barn Computer Club
319 Main St.
Suite #2
Salinas, CA 93901
757-0788
S. Mark Vanderbilt

Humboldt Commodore Group
P.O. Box 570
Arcata, CA 95521
R. Turner

Napa Valley Commodore Computer Club
c/o Liberty Computerware
2680 Jefferson St.
Napa, CA 94558
(707) 252-6281
Mick Winter
1st & 3rd Mon. of month

S.D. East County C-64 User Group
6353 Lake Apopka Place
San Diego, CA 92119
(619) 698-7814
Linda Schwartz

COLORADO

VICKIMPET Users Group
4 Waring Lane, Greenwood Village
Littleton, CO 80121
Contact: Louis Roehrs

Colorado Commodore Computer Club
2187 S. Golden Ct.
Denver, CO 80227
986-0577
Jack Moss
Meet: 2nd Wed.

CONNECTICUT

John F. Garbarino
Skiff Lane Masons Island
Mystic, CT 06355
(203) 536-9789

Commodore User Club
Wethersfield High School
411 Wolcott Hill Road
Wethersfield, CT 06109
Contact: Daniel G. Spaneas

VIC Users Club
c/o Edward Barszczewski
22 Tunxis Road
West Hartford, CT 06107

New London County Commodore Club
Doolittle Road
Preston, CT 06360
Contact: Dr. Walter Doolittle

FLORIDA

Jacksonville Area PET Society
401 Monument Road, #177
Jacksonville, FL 32211

Richard Prestien
6278 SW 14th Street
Miami, FL 33144

South Florida PET Users Group
Dave Young
7170 S.W. 11th
West Hollywood, FL 33023
(305) 987-6982

VIC Users Club
c/o Ray Thigpen
4071 Edgewater Drive
Orlando, FL 32804

PETs and Friends
129 NE 44 St.
Miami, FL 33137
Richard Plumer

Sun Coast VICs
P.O. Box 1042
Indian Rocks Beach, FL 33535
Mark Weddell

Bay Commodore Users Group
c/o Gulf Coast Computer Exchange
241 N. Tyndall Pkwy.
P.O. Box 6215
Panama City, FL 32401
(904) 785-6441
Richard Scofield

Gainesville Commodore Users Club
3604-20A SW 31st Dr.
Gainesville, FL 32608
Louis Wallace

64 Users Group
P.O. Box 561689
Miami, FL 33156
(305) 274-3501
Eydie Sloane

Brandon Users Group
108 Anglewood Dr.
Brandon, FL 33511
(813) 685-5138
Paul Daugherty

Commodore 64/VIC 20 User Group
Martin Marietta Aerospace
P.O. Box 5837, MP 142
Orlando, FL 32855
(305) 352-3252/2266
Mr. Earl Preston

Brandon Commodore Users Group
414 E. Lumsden Rd.
Brandon, FL 33511

Gainesville Commodore Users Group
Santa Fe Community College
Gainesville, FL 32602
James E. Birdsell

Commodore Computer Club
P.O. Box 21138
St. Petersburg, FL 33742

GEORGIA

VIC Educators Users Group
Cherokee County Schools
110 Academy St.
Canton, GA 30114
Dr. Al Evans

Bldg. 68, FLETC
Glynco, GA 31524
Richard L. Young

VIC-tims
P.O. Box 467052
Atlanta, GA 30346
(404) 922-7088
Eric Ellison

HAWAII

Commodore Users Group of Honolulu
c/o PSH
824 Bannister St.
Honolulu, HI
(808) 848-2088
3rd Fri. every month

IDAHO

GHS Computer Club
c/o Grangeville High School
910 S. D St.
Grangeville, ID 83530
Don Kissinger

S.R.H.S. Computer Club
c/o Salmon River H.S.
Riggins, ID 83549
Barney Foster

Commodore Users
548 E. Center
Pocatello, ID 83201
(208) 233-0670
Leroy Jones

Eagle Rock Commodore Users Group
900 S. Emerson
Idaho Falls, ID 83401
Nancy J. Picker

ILLINOIS

Shelly Wernikoff
2731 N. Milwaukee
Avenue
Chicago, IL 60647

VIC 20/64 Users Support
Group
c/o David R. Tarvin
114 S. Clark Street
Pana, IL 62557
(217) 562-4568

Central Illinois PET User
Group
635 Maple
Mt. Zion, IL 62549
(217) 864-5320
Contact: Jim Oldfield

ASM/TED User Group
200 S. Century
Rantoul, IL 61866
(217) 893-4577
Contact: Brant Anderson

PET VIC Club (PVC)
40 S. Lincoln
Mundelein, IL 60060
Contact: Paul Schmidt,
President

Rockford Area PET Users
Group
1608 Benton Street
Rockford, IL 61107

Commodore Users Club
1707 East Main St.
Olney, IL 62450
Contact: David E. Lawless

VIC Chicago Club
3822 N. Bell Ave.
Chicago, IL 60618
John L. Rosengarten

Chicago Commodore 64
Users & Exchange Group
P.O. Box 14233
Chicago, IL 60614
Jim Robinson

Fox Valley PET Users
Group
833 Willow St.
Lake in the Hills, IL 60102
(312) 658-7321
Art DeKneef

The Commodore 64 Users
Group
4200 Commerce Ct., Suite 100
Lisle, IL 60532
(312) 369-6525
Gus Pagnotta

Oak Lawn Commodore Users Group
The Computer Store
11004 S. Cicero Ave.
Oak Lawn, IL 60453
(312) 499-1300
Bob Hughes

INDIANA

PET/64 Users
10136 E. 96th St.
Indianapolis, IN 46256
(317) 842-6353
Jerry Brinson

Cardinal Sales
6225 Coffman Road
Indianapolis, IN 46268
(317) 298-9650
Contact: Carol Wheeler

CHUG (Commodore
Hardware Users Group)
12104 Meadow Lane
Oaklandon, IN 46236
Contact: Ted Powell

VIC Indy Club
P.O. Box 11543
Indianapolis, IN 46201
(317) 898-8023
Ken Ralston

Northern Indiana
Commodore Enthusiasts
927 S. 26th St.
South Bend, IN 46615
Eric R. Bean

Commodore Users Group
1020 Michigan Ave.
Logansport, IN 46947
(219) 722-5205
Mark Bender

Computer Workshop VIC 20/64 Club
282 S. 600 W.
Hebron, IN 46341
(219) 988-4535
Mary O'Bringer

The National Science Clubs of America
Commodore Users Division
7704 Taft St.
Merrillville, IN 46410
Brian Lapley or Tom Vlasic

East Central Indiana VIC User Group
Rural Route #2
Portland, IN 47371
Stephen Erwin

National VIC 20 Program Exchange
102 Hickory Court
Portland, IN 47371
(219) 726-4202
Stephen Erwin

IOWA

Commodore User Group
114 8th St.
Ames, IA 50010

Quad City Commodore Club
1721 Grant St.
Bettendorf, IA 52722
(319) 355-2641
John Yigas

Commodore Users Group
965 2nd St.
Marion, IA 52302
(319) 377-5506
Vern Rotert
3rd Sun. of month

Siouxland Commodore Club
2700 Sheridan St.
Sioux City, IA 51104
(712) 258-7903
Gary Johnson
1st & 3rd Monday of month

421 W. 6th St.
Waterloo, IA 50702
(319) 232-1062
Frederick Volker

Commodore Computer Users
Group of Iowa
Box 3140
Des Moines, IA 50316
(515) 263-0963 or (515) 287-1378
Laura Miller

KANSAS

Wichita Area PET
Users Group
2231 Bullinger
Wichita, KS 67204
(316) 838-0518
Contact: Mel Zandler

Kansas Commodore
Computer Club
101 S. Burch
Olathe, KS 66061
Contact: Paul B. Howard

Commodore Users Group
6050 S. 183 St. West
Viola, KS 67149
Walter Lounsbery

KENTUCKY

VIC Connection
1010 S. Elm
Henderson, KY 42420
Jim Kemp

LOUISIANA

Franklin Parish Computer
Club
#3 Fair Ave.
Winnisboro, LA 71295
James D. Mays, Sr.

NOVA
917 Gordon St.
New Orleans, LA 70117
(504) 948-7643
Kenneth McGruder, Sr.

VIC 20 Users Group
5064 Bowdon St.
Marrero, LA 70072
(504) 341-5305
Wayne D. Lowery, R.N.

MARYLAND

Assoc. of Personal
Computer Users
5014 Rodman Road
Bethesda, MD 20016

Blue TUSK
700 East Joppa Road
Baltimore, MD 21204
Contact: Jim Hauff

House of Commodore
8835 Satyr Hill Road
Baltimore, MD 21234
Contact: Ernest J. Fischer

Long Lines Computer Club
323 N. Charles St., Rm. 201
Baltimore, MD 21201
Gene Moff

VIC & 64 Users Group
The Boyds Connection
21000 Clarksburg Rd.
Boyds, MD 20841
(301) 428-3174
Tom DeReggi

VIC 20 Users Group
23 Coventry Lane
Hagerstown, MD 21740
Joseph Rutkowski

Hagerstown Users Group
1201-B Marshall St.
Hagerstown, MD 21740
(301) 790-0968
Greg Stewart
1st & 3rd Friday of month 6:30 p.m.

Rockville VIC/64 Users Group
13013 Evanstown St.
Rockville, MD 20853
(301) 946-1564
Meryle or Tom Pounds

MASSACHUSETTS

Eastern Massachusetts
VIC Users Group
c/o Frank Ordway
7 Flagg Road
Marlboro, MA 02173

VIC Users Group
c/o Ilene Hoffman-Sholar
193 Garden St.
Needham, MA 02192

Commodore Users Club
Stoughton High School
Stoughton, MA 02072
Contact: Mike Lennon

Berkshire PET Lovers
CBM Users Group
Taconic High
Pittsfield, MA 01201

The Boston Computer
Society
Three Center Plaza
Boston, MA 02108
(617) 367-8080
Mary E. McCann

VIC Interface Club
c/o Procter & Gamble Inst. Shop
780 Washington St.
Quincy, MA 02169
C. Gary Hall

Masspet Commodore Users Group
P.O. Box 307
East Taunton, MA 02718
David Rogers

Raytheon Commodore Users Group
Raytheon Company
Hartwell Rd. GRA-6
Bedford, MA 01730
John Rudy

Commodore 64 Users
Group of The Berkshires
184 Highland Ave.
Pittsfield, MA 01201
Ed Rucinski

MICHIGAN

David Liem
14361 Warwick Street
Detroit, MI 48223

VIC Users Club
University of Michigan
School of Public Health
Ann Arbor, MI 48109
Contact: John Gannon

Commodore User Club
32303 Columbus Drive
Warren, MI 48093
Contact: Robert Steinbrecher

Commodore Users Group
c/o Family Computer
3947 W. 12 Mile Rd.
Berkley, MI 48072

W. Michigan VIC 20-64 Users
1311 Portland NE
Grand Rapids, MI 49505
(616) 459-7578
Jim D'Haem

VIC for Business
6027 Orchard Ct.
Lansing, MI 48910
Mike Marotta

South Computer Club
South Jr. High School
45201 Owen
Belleville, MI 48111
Ronald Ruppert

Commodore Users Group
c/o Eaton Rapids Medical Clinic
101 Spicerville Hwy.
Eaton Rapids, MI 48827
Albert Meinke III, M.D.

South East Michigan Pet Users Group
Box 214
Farmington, MI 48024
Norm Eisenberg

Commodore Computer Club
H. Dow High School, Rm #226
Midland, MI 48640
(517) 835-5130
John Walley
9:30 p.m. Sept/May

VIC, 64, PET Users Group
8439 Arlis Rd.
Union Lake, MI 48085
363-8539
Bert Searing

MINNESOTA

MUPET (Minnesota Users of PET)
P.O. Box 179
Annandale, MN 55302
c/o Jon T. Minerich

Twin Cities Commodore
Computer Club
6623 Ives Lane
Maple Grove, MN 55369
(612) 424-2425
Contact: Rollie Schmidt

MISSOURI

KCPUG
5214 Blue Ridge Boulevard
Kansas City, MO 64133
Contact: Rick West
(816) 356-2382

PET SET Club of St. Louis
633 Bent Oak Drive
Lake St. Louis, MO 63367
(314) 625-2701 or 625-4576
Tony Ott

VIC INFONET
P.O. Box 1069
Branson, MO 65616
(417) 334-6099
Jory Sherman

Worth County PET Users
Group
Grant City, MO
(816) 564-3551
David Hardy

Mid-Missouri Commodore Club
1804 Vandiver Dr.
Columbia, MO 65201
(314) 474-4511
Phil Bishop

MONTANA

Powder River
Computer Club
Powder River County
High School
Broadus, MT 59317
Contact: Jim Sampson

Commodore User Club
1109 West Broadway
Butte, MT 59701
Contact: Mike McCarthy

NEVADA

Las Vegas PET Users
4884 Iron Avenue
Las Vegas, NV 89110

NEW JERSEY

Amateur Computer Group
18 Alpine Drive
Wayne, NJ 07470

Somerset Users Club
49 Marcy Street
Somerset, NJ 08873
Contact: Robert Holzer

Educators Advisory
P.O. Box 186
Medford, NJ 08055
(609) 953-1200
John Handfield

VIC-TIMES
46 Wayne Street
Edison, NJ 08817
Thomas R. Molnar

VIC 20 User Group
67 Distler Ave.
W. Caldwell, NJ 07006
(201) 284-2281
G. M. Amin

VIC Software Development Club
77 Fomalhaut Ave.
Sewell, NJ 08080
H. P. Rosenberg

ACGNJ PET/VIC/CBM
User Group
30 Riverview Terr.
Belle Mead, NJ 08502
(201) 359-3862
J. M. Pylka

South Jersey Commodore Computer
Users Club
46-B Monroe Park
Maple Shade, NJ 08052
(609) 667-9758
Mark Orthner
2nd Fri. of month

NEW HAMPSHIRE

Northern New England
Computer Society
P.O. Box 69
Berlin, NH 03570

TBH VIC-NICs
P.O. Box 981
Salem, NH 03079

NEW MEXICO

Commodore Users Group
6212 Karlson, NE
Albuquerque, NM 87113
(505) 821-5812
Danny Byrne

NEW YORK

Capital District 64/VIC 20 Users Group
363 Hamilton St.
Albany, NY 12210
(518) 436-1190
Bill Pizer

Long Island PET Society
Ralph Bressler
Harborfields HS
Taylor Avenue
Greenlawn, NY 11740

PET User Club
of Westchester
P.O. Box 1280
White Plains, NY 10602
Contact: Ben Meyer

LIVE (Long Island
VIC Enthusiasts)
17 Picadilly Road
Great Neck, NY 11023
Contact: Arnold Friedman

Commodore Masters
25 Croton Ave.
Staten Island, NY 10301
Contact: Stephen Farkouh

VIC Users Club
76 Radford St.
Staten Island, NY 10314
Contact: Michael Frantz

Rockland County Commodore
Users Group
c/o Ross Garber
14 Hillside Court
Suffern, NY 10901
(914) 354-7439

West Chester County VIC
Users Group
P.O. Box 146
Pelham, NY 10552
Joe Brown

SPUG
4782 Boston Post Rd.
Pelham, NY 10803
Paul Skipski

VIC 20 User Club
151-28 22nd Ave.
Whitestone, NY 11357
Jean F. Coppola

VIC 20 User Club
339 Park Ave.
Babylon, NY 11702
(516) 669-9126
Gary Overman

VIC User Group
1250 Ocean Ave.
Brooklyn, NY 11230
(212) 859-3030
Dr. Levitt

L&M Computer Club
VIC 20 & 64
4 Clinton St.
Tully, NY 13159
(315) 696-8904
Dick Mickelson

Commodore Users Group
1 Corwin Pl.
Lake Katrine, NY 12449
J. Richard Wright

8*8 Enthusiasts
P.O. Box 28 Rhodes Rd.
Apalachin, NY 13732
Keith Merrill

VIC 20/Commodore 64
Users Group
31 Maple Dr.
Lindenhurst, NY 11757
(516) 957-1512
Pete Lobol

VIC Information Exchange
Club
336 W. 23 St.
Deer Park, NY 11729
Tom Schlegel
SASE & phone please

New York Commodore
Users Group
380 Riverside Dr., 7Q
New York, NY 10025
(212) 566-6250
Ben Tunkelang

Parsippany Computer Group
51 Ferncliff Rd.
Morris Plains, NJ 07950
(201) 267-5231
Bob Searing

Hudson Valley Commodore Club
1 Manor Dr.
Woodstock, NY 12498
F.S. Goh
1st Wednesday of month

LIVICS (Long Island VIC Society)
20 Spyglass Lane
East Setauket, NY 11733
(516) 751-7844
Lawrence Stefani

VIC Users Group
c/o Stoney Brook Learning Center
1424 Stoney Brook Rd.
Stoney Brook, NY 11790
(516) 751-1719
Robert Wurtzel

user groups

NORTH CAROLINA

Amateur Radio PET Users Group
P.O. Box 30694
Raleigh, NC 27622
Contact: Hank Roth

VIC Users Club
c/o David C. Fonenberry
Route 3, Box 351
Lincolnton, NC 28092

Microcomputer Users Club
Box 17142 Bethabara Sta.
Winston-Salem, NC 27116
Joel D. Brown

VIC Users Club
Rt. 11, Box 686
Hickory, NC 28601
Tim Gromlovits

OHIO

Dayton Area PET
User Group
933 Livingston Drive
Xenia, OH 45385
B. Worby, President
(513) 848-2065
J. Watson, Secretary
(513) 372-2052

Central Ohio PET
Users Group
107 S. Westmoor Avenue
Columbus, OH 43204
(614) 274-6451
Contact: Philip H. Lynch

Toledo PETS
734 Donna Drive
Temperance, MI 48182
Contact: Gerald Carter,
President

Chillicothe Commodore
Users Group
P.O. Box 211
Chillicothe, OH 45601
William A. Chaney

Licking County 64 Users Group
323 Schuler St.
Newark, OH 43055
(614) 345-1327

11433 Pearl Rd.
Strongsville, OH 44136
Paul M. Warner

OKLAHOMA

Southwest Oklahoma
Computer Club
P.O. Box 6646
Lawton, OK 73504
Garry Lee Crowell
1:30 1st Sunday at
Lawton City Library

Tulsa Area Commodore Users Group
Tulsa Computer Society
P.O. Box 15238
Tulsa, OK 74112
Annette Hinshaw

Commodore Oklahoma Users Club
4000 NW 14th St.
Oklahoma City, OK 73107
(405) 943-1370
Stanley B. Dow

OREGON

NW PET Users Group
John F. Jones
2134 N.E. 45th Avenue
Portland, OR 97213

PENNSYLVANIA

PET User Group
Gene Beals
P.O. Box 371
Montgomeryville, PA 18936

Penn Conference Computer Club
c/o Penn Conference of SDA
720 Museum Road
Reading, PA 19611
Contact: Dan R. Knepp

PACS PET Users Group
20th & Olney Streets
Philadelphia, PA 19120

Glen Schwartz
807 Avon
Philadelphia, PA 19116

Gene Planchak
4820 Anne Lane
Sharpsville, PA 15150
(412) 962-9682

PPG (Pittsburgh PET Group)
c/o Joel A. Casar, DMD
2015 Garrick Drive
Pittsburgh, PA 15235
(412) 371-2882

Westmoreland Commodore
Users Club
c/o DJ & Son Electronics
Colonial Plaza
Latrobe, PA 15650
Jim Mathers

COMPSTARS
440 Manatawny St.
Pottstown, PA 19464
Larry Shupinski, Jr.
Meet at Audio Video
Junction

Commodore Users Club
3021 Ben Venue Dr.
Greensburg, PA 15601
(412) 836-2224
Jim Mathers

VIC 20 Programmers, Inc.
c/o Watson Woods
115 Old Spring Rd.
Coatesville, PA 19320
Robert Gougher

G.R.C. User Club
300 Whitten Hollow Rd.
New Kensington, PA 15068
Bill Bolt

NADC Commodore Users Club
248 Oakdale Ave.
Horsham, PA 19044
Norman McCrary

CACC (Capitol Area Commodore Club)
134 College Hill Rd.
Enola, PA 17025
(717) 732-2123
Lewis Buttery
Union Deposit Mall at 7 p.m.

G/C Computer Owners Group
P.O. Box 1498
Reading, PA 19607
Jo Lambert
(215) 775-2600, ex 6472

PUERTO RICO

CUG of Puerto Rico
RFD #1, Box 13
San Juan, PR 00914
Ken Burch

VIC 20 User Group
655 Hernandez St.
Miramar, PR 00907
Robert Morales, Jr.

RHODE ISLAND

Irving B. Silverman, CPA
160 Taunton Ave.
E. Providence, RI 02914
Contact: Michelle Chavanne

Newport VIC/64 Users
10 Maitland Ct.
Newport, RI 02840
(401) 849-2684
Dr. Matt McConeghy

The VIC 20 Users Club
Warwick, RI 02886
Tom Davey

SOUTH CAROLINA

Beauford Technical College
100 S. Ribaut Rd.
Beauford, SC 29902
Dean of Instruction

SOUTH DAKOTA

PET User Group
515 South Duff
Mitchell, SD 57301
(605) 996-8277
Contact: Jim Dallas

VIC/64 Users Club
203 E. Sioux Ave.
Pierre, SD 57501
(605) 224-4863
Larry Lundeen

TENNESSEE

River City Computer
Hobbyists
Memphis, TN
1st Mon. at Main Library

Nashville Commodore Users Group
P.O. Box 121282
Nashville, TN 37212
3rd Thurs. at Cumberland Mus

Commodore User Club
Metro Computer Center
1800 Dayton Blvd.
Chattanooga, TN 37405
Mondays 7:30 pm

Metro-Knoxville 64 Users Club
7405 Oxmoor Rd., Rt. #20
Knoxville, TN 37921
(615) 938-3773
Ed Pritchard

TEXAS

SCOPE
1020 Summit Circle
Carrolton, TX 75006

PET Users
2001 Bryan Tower
Suite 3800
Dallas, TX 75201

Larry Williams
P.O. Box 652
San Antonio, TX 78293

PET User Group
John Bowen
Texas A & M
Microcomputer Club
Texas A & M, TX

CHUG (Commodore Houston
Users Group)
8738 Wildforest
Houston, TX 77088
(713) 999-3650
Contact: John Walker

Corpus Christi Commodores
3650 Topeka St.
Corpus Christi, TX 78411
(512) 852-7665
Bob McKelvy

Commodore Users Group
5326 Cameron Rd.
Austin, TX 78723
(512) 459-1220
Dr. Jerry D. Frazee

VIC Users Group
3817 64th Dr.
Lubbock, TX 79413

Southeast Houston VIC
Users Group
11423 Kirk Valley Dr.
Houston, TX 77089
(713) 481-6653

64 Users Group
2421 Midnight Circle
Plano, TX 75075
S.G. Grodin

UTAH

Utah PUG
Jack Fleck
2236 Washington Blvd.
Ogden, UT 84401

The Commodore Users
Club
742 Taylor Avenue
Ogden, UT 84404
Contact: Todd Woods Kap,
President
David J. Shreeve,
Vice President

The VIClic
799 Ponderosa Drive
Sandy, UT 84070
Contact: Steve Graham

VIC 20 Users
324 N. 300 W.
Smithfield, UT 84335
Dave DeCorso

Northern Utah VIC & 64
Users Group
P.O. Box 533
Garland, UT 84312
David Sanders

The Utah Commodore Users Group
652 West 700 North
Clearfield, UT 84015
(801) 776-3950
Rodney Keller or Richard Brenchly

VIRGINIA

Northern VA PET Users
Bob Karpen
2045 Eakins Court
Reston, VA 22091
(803) 860-9116

VIC Users Group
Rt. 2, Box 180
Lynchburg, VA 24501
Contact: Dick Rossignol

VIC Users Group
c/o Donnie L. Thompson
1502 Harvard Rd.
Richmond, VA 23226

Dale City Commodore
User Group
P.O. Box 2004
Dale City, VA 22193
(703) 680-2270
James Hogler

Tidewater Commodore
Users Group
4917 Westgrove Rd.
Virginia Beach, VA 23455
Fred Monson

Fredericksburg Area
Computer Enthusiasts
P.O. Box 324
Locust Grove, VA 22508
(703) 972-7195
Michael Parker

Commonwealth 20/64
Users Group
1773 Wainwright Dr.
Reston, VA 22090
(703) 471-6325
Tal Carawan, Jr.

VIC 20 Victims
4301 Columbia Pike #410
Arlington, VA 22204
(703) 920-0513
Mike Spengel

Peninsula Commodore 64 Users Group
124 Burnham Place
Newport News, VA 23606
(804) 595-7315
Richard G. Wilmoth

WASHINGTON

NW PET Users Group
2565 Dexter N. 3203
Seattle, WA 98109
Contact: Richard Ball

PET Users Group
c/o Kenneth Tong
1800 Taylor Ave. N102
Seattle, WA 98102

Whidbey Island Commodore
Computer Club
947 N. Burroughs Ave.
Oak Harbor, WA 98277
Michael D. Clark

Central Washington
Commodore Users Group
1222 S. 1st St.
Yakima, WA 98902
Tim McElroy

Blue Mountain Commodore
Users Club
667 Canary Dr.
Walla Walla, WA 99362
(509) 525-5452
Keith Rodue

WEST VIRGINIA

Personal Computer Club
P.O. Box 1301
Charleston, WV 25325
Cam Cravens

WISCONSIN

Sewpus
c/o Theodore J. Polozynski
P.O. Box 21851
Milwaukee, WI 53221

Waukesha Area Commodore
User Group (WACUG)
256½ W. Broadway
Waukesha, WI 53186
Contact: Walter Sadler
(414) 547-9391

Commodore User Group
1130 Elm Grove St.
Elm Grove, WI 53122
Tony Hunter

Commodore 64 Software
Exchange Group
P.O. Box 224
Oregon, WI 53575
E. J. Rosenberg

C.L.U.B. 84
6156 Douglas Ave.
Caledonia, WI 53108
(414) 835-4645 pm
Jack White
2nd Sat every month 10:00 am

VIC-20 & 64 User Group
522 West Bergen Dr.
Milwaukee, WI 53217
(414) 476-8125
Mr. Wachtl

CANADA

Toronto PET
Users Group
381 Lawrence Ave. West
Toronto, Ontario, Canada
M5M 1B9
(416) 782-9252
Contact: Chris Bennett

PET Users Club
c/o Mr. Brown
Valley Heights Secondary Schoo
Box 159
Langton, Ont. N0E 1G0

Vancouver PET Users Group
P.O. Box 91164
West Vancouver, British
Columbia
Canada V7V 3N6

CCCC (Canadian
Commodore Computer Club)
c/o Strictly Commodore
47 Coachwood Place
Calgary, Alberta, Canada
T3H 1E1
Contact: Roger Olanson

W.P.U.G.
9-300 Enniskillen Ave.
Winnipeg, Manitoba R2V 0H9
Larry Neufeld

VIC-TIMS
2-830 Helena St.
Trail, British Columbia
V1R 3X2
(604) 368-9970
Greg Goss

Arva Hackers
Medway High School
Arva, Ontario N0M 1C0
D. Lerch

Nova Scotia Commodore
Computer Users Group
66 Landrace Cres.
Dartmouth, N.S. B2W 2P9
Andrew Cornwall

Bonnyville VIC Cursors
Box 2100
Bonnyville, Alberta T0A 0L0
(403) 826-3992
Ed Wittchen

FINLAND

VIC-Club in Helsinki
c/o Matti Aarnio
Linnustajankj 2B7
SF-02940 ESP00 94
Finland

KOREA

Commodore Users Club
K.P.O. Box 1437
Seoul, Korea
Contact: S. K. Cha

MEXICO

Asociacion De Usarios
Commodore
c/o Alejandro Lopez
Arechiga
Holbein 174-6° Piso
Mexico 18, D.F.

Club de Usarios Commodore
Sigma del Norte
Mol del Valle, Local 44
Garza Garcia, N.L. 66220

NEW ZEALAND

Commodore Users Group
Meet at VHF Clubrooms
Hazel Ave.
Mount Roskill
3rd Wed. of month, 7:30 pm
Roger Altena 278-5262

Nelson VIC Users Group
c/o P.O. Box 860
Nelson, New Zealand
Peter Archer

E.R. Kennedy
c/o New Zealand Synthetic
Fuels Corp. Ltd.
Private Bag
New Plymouth

NORWAY

VIC Club of Norway
Nedre Bankegt 10,
1750 Halden
Norway

UNITED KINGDOM

North London Hobby
Computer Club
Dept. of Electronics &
Communications
Engineering
The Polytechnic of North
London
Holloway Rd.
London N7 8DB

Croydon Microcomputer Club
111 Selhurst R.
Selhurst, London SE25 6LH
01-653-3207
Vernon Gifford

User Bulletin Board

User Groups Forming:

CALIFORNIA

Pulare County Area
Contact Alan Landon
1555 E. Sonora
Pulare, CA 93274
(209) 688-6832

FLORIDA

VIC 20 Group
Contact Frank H. Topping
1859 Neptune Drive
Englewood, FL 33533

MARYLAND

VIC 20 Group
Contact Robert F. Buck, Jr.
814 Brentwood Manor
Sedalia, MD 65301

Classified

ATTENTION VIC 20™ OWNERS!
Educational Programs at
Bargain Prices!
Write to Victoria Williams, Ph.D.,
Palos Verdes Learning Center,
716 Yarmouth Road, #203P,
Palos Verdes Estates, CA 90274.

Fun, Games and Educational
Programs under $5.00. For free
catalog write to: SOFT 4 YOU,
P.O. Box 3254, Reston, VA 22090.

Micrographics™

- Custom and proprietary software for the Commodore 64™
- Specialists in image processing and graphics design

Micrographics Corp.
5301 Wakefield Road
Bethesda MD. 20816
(301) 951-0418
Commodore 64 is a trademark of
Commodore Electronics Ltd.

that does not compute...

"Cursor Input Program"

March, 1983

In lines 220 and 230 of the program listing, the cursor right commands should be cursor left. Also, the program listing as shown in that issue used graphic characters from PET/CBM, so VIC 20 and Commodore 64 owners may not have realized that the program will also run on their computers. For the benefit of VIC and 64 users, the reverse graphic characters translate as follows:

rvs q is cursor down
rvs Q is cursor up
rvs r is ctrl rvs on
rvs R is ctrl rvs off
rvs S is shift clr/home

"Studying Complex Rhythms on the Commodore 64"

March, 1983

According to the last paragraph on page 76, the 64 allows up to four simultaneous channels of sound. That's not quite correct. The sound chip in the Commodore 64 generates three channels of sound. A fourth channel is available by using the external input capabilities of the chip.

Also, the note table on page 80 is correct only for the very early versions of the 64. The computers presently being distributed have a clock frequency of about 1.02 MHz. The correct note table appears in the *Commodore 64 Programmer's Reference Guide.*

The VIC Magician "PEEKing Keys to Control Actions"

March, 1983

The formulas for left and right columns on page 48 need revision. They should read as follows:

L=INT(23*RND(1))
LF=7680+(22*L)
and
R=INT(23*RND(1))
RT=7701+(22*R)

These formulas generate a random number between 0 and 22 (the article said they generate a number between 1 and 23).

Also, on page 51, second column, line 95 should read:
95 IFPEEK(E−1)=90THEN20

"Seeing RTTY on the VIC"

December 1982/January 1983

We took some liberties with the program listings on page 35 that may have confused our readers. Bruce Cameron, one of the authors, submits the following corrections to straighten it all out:

ASCII Program

```
5 REM "ASCII"
10 OPEN2,2,0,CHR$(160+1)+CHR$(160)
20 GET#2,C$
30 IFC$>=" " AND C$<="↑" OR C$=CHR$(13)
   THENPRINTC$;
 40 GOTO20
```

Baudot Letters Only

```
5 REM"BAUDOT LTRS"
10 OPEN2,2,0,CHR$(96+1)+CHR$(0)
20 T$="E A SIU"+CHR$(13)+
   "DRJNFCKTZLWHYPQOBG MXV "
30 GET#2,C$:IFC$=""THEN30
40 T=ASC(C$)
50 IFT>0THENPRINTMID$(T$,T,1);
60 GOTO30
```

Full Baudot

```
5 REM "FULL BAUDOT"
10 OPEN2,2,0,CHR$(96+1)+CHR$(0)
20 LS=-1
30 LF$=CHR$(10)
40 CR$=CHR$(13)
50 L$="E"+LF$+"A SIU"+CR$+
   "DRJNFCKTZLWHYPQOBG*MXV*"
60 F$="3"+LF$+"- ´87"+CR$+"$4´,!:
   (5´)2#6019?&*./:*"
100 GET#2,C$:IFC$=""THEN160
110 C=ASC(C$):IFC<1ORC>31THEN100
120 IFLSTHENC$=MID$(L$,C,1)
130 IFNOTLSTHENC$=MID$(F$,C,1)
140 IFC$<>"*"THENPRINTC$;:GOTO160
```

```
150 LS=(C=31)
160 GETA$:IFA$=""THEN100
170 IFA$="L"THENLS=-1
175 IFA$="F"THENLS=0
180 GOTO100
```

"S.O.S. Revisited"

December 1982/January 1983

To have the program on page 56 loop infinitely, change line 230 to read:

230 IF A$(I)<A$(I+G)ORA$(I)=A$(I+G)GOTO280

Thanks to R. Cicchinelli of Pittsburgh, Pennsylvania, for that one. **C**

new books

From Osborne/ McGraw Hill

2600 Tenth Street
Berkeley, California 94710

VisiCalc® Made Easy by David M. Castlewitz. A step-by-step tutorial divided into three parts. The first part introduces the basic skills needed to build a worksheet. The second examines the commands used to change and edit a worksheet. And the third covers advanced uses and special tricks that extend the capabilities of the VisiCalc programs.

From Hayden Book Company

50 Essex Street
Rochelle Park, New Jersey 07662

Software Toolkit for Microcomputers by Max Schindler. An edited compilation of articles from *Electronic Design* magazine that provides a comprehensive discussion of how to use high-level languages and operating systems to speed up software design.

CP/M® Revealed by Jack Dennon. Describes in detail the full potential of CP/M, including the system's technical aspects, the system manager, the input/output driver package and the system's data structure. The book includes CP/M utilities and other essential information for using the system effectively.

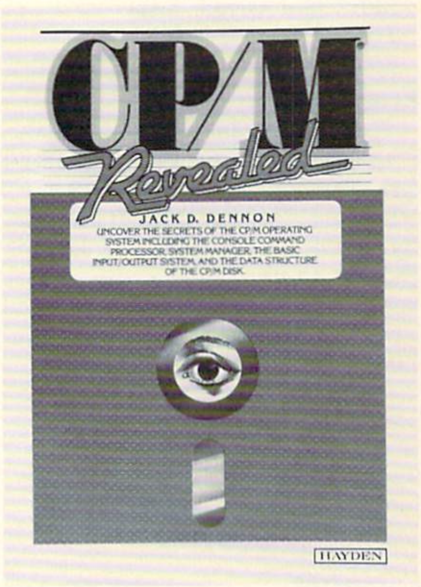
I Speak BASIC to my VIC by Aubrey Jones. A computer literacy course that introduces students to BASIC programming and operation of the VIC 20. Includes a teacher's manual, student text and exam set.

From the dilithium Press

11000 S.W. 11th Street, Suite E
Beaverton, Oregon 97005

PET/CBM BASICS by D.J. David. A programming tutorial structured so the reader progressively learns the elements of programming in BASIC on the PET and CBM computers. Includes information on graphics, curves and animation.

VisiCalc is a registered trademark of VisiCorp
CP/M is a registered trademark of Digital Research, Inc.


CP/M
Revealed
JACK D. DENNON
UNCOVER THE SECRETS OF THE CP/M OPERATING SYSTEM INCLUDING THE CONSOLE COMMAND PROCESSOR, SYSTEM MANAGER, THE BASIC INPUT/OUTPUT SYSTEM, AND THE DATA STRUCTURE OF THE CP/M DISK
HAYDEN

new products

The following information is taken from product announcements sent to us by independent manufacturers and is provided to help keep our readers abreast of developments in the field. Commodore does not endorse any of the products listed, has not tested them and cannot vouch for their availability. If you have any problems with any of the products listed here, please write to us.

Company:
Quick Return Co.
31912 36th Avenue, S.W.
Federal Way, WA 98003
206-927-8980

Product:
Federal Individual Tax Preparation Program—for the Commodore 64 with disk drive and printer. Form 1040 is printed for reproduction with the overlay included. Supporting schedules and forms are produced on the printer to IRS specifications. The 1982 version included 28 forms and schedules. The 1983 program will be upgraded to include more. Designed for use by professional tax return preparers. Not all state returns may be available for the 1983 program.
Price: Federal return $225; Average cost with state return $265

Company:
Galactic Software
P.O. Box 10516
San Jose, CA 95157
408-247-4434

Product:
Mailing list—for the VIC 20 with 16K expander. Each record contains name, address and three comment fields. Capability to alphabetize upon entry, sort and search all fields, print labels or print complete records. Completely menu driven.
Price: Tape $25.95; disk $27.95

Company:
Home Computer Corporation
154 Heard Road
Kathleen, GA 31047
912-987-0235

Product:
Numeric keypad—for the VIC 20 and Commodore 64. A 24-key array to aid in the entry of numerical data in business and machine language applications. Available in standard or hexadecimal versions. Easy to install.
Price: $79.95

Company:
Programmers Guild
P.O. Box 220
LaCrescent, MN 55947

Product:
Three educational programs—for the VIC 20. *Math Escape* aids in learning addition, subtraction, multiplication and division. Unlimited levels of difficulty. *Sea Word* has the student unscramble the graded words to defeat the sea serpent. *Birthday Card* and *Story Time* on one tape. First have a party, complete with song, cake and more. Then help the computer write stories.
Price: Math Escape $14.95; Sea Word and Birthday Card/Story Time $9.95

Company:
Fabtronics
51 Quarry Street
Brockport, NY 14420
716-637-6371

Product:
Utility File—for the VIC 20 (with 3K expansion) and Commodore 64. A data processing program to calculate, display, file and print out data on utility consumption (including electric, water, gas, oil and propane) for designated unit numbers. Can also project cost for any number of days. For residential or commercial use.
Price: Tape $17.95; disk $20.95. Add $2.00 shipping and handling.

Micron Eye Vision System

Company:
Micron Technology, Inc.
2805 East Columbia Road
Boise, ID 83706
208-383-4000

Product:
MicronEye vision system—for the Commodore 64. Transmits images to the computer's memory, enabling graphics display, image analysis and image storage to disk. Possible applications include program animation, security, automated process control, digitizing, pattern analysis, robot vision and text recognition. Capable of 256 × 128 resolution and operating speeds of up to 15 frames per second. Sample programs, driver routines and documentation included.
Price: Contact company

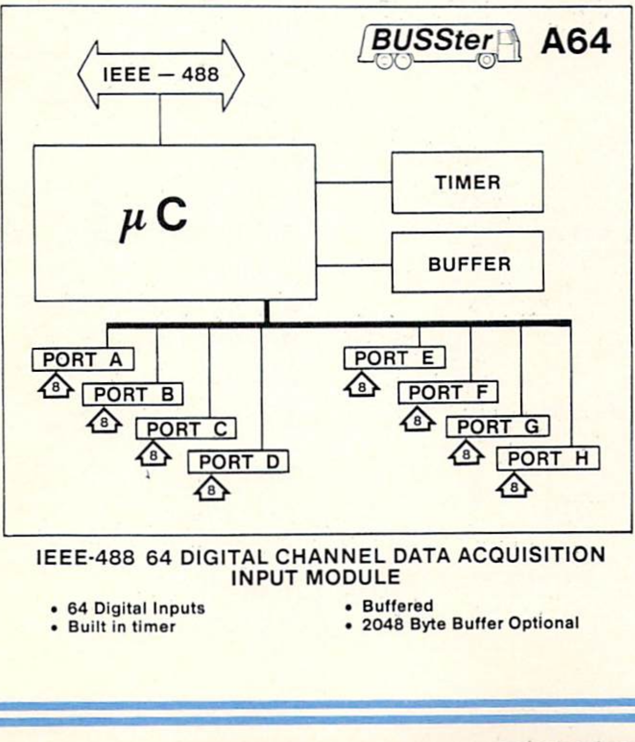
Company:
Midwest Software
Box 214
Farmington, MI 48024
313-477-0897

Product:
New useful programs—for PET/CBM

and Commodore 64. *Datalog* lets you define up to 12 fields and creates up to 1000 records on a 4040 disk. Interfaces with popular word processors to print form letters or labels. Multiple sort and search capabilities. Requires 32K. *Date Due* manages overdue items in libraries. Prints reports by name, date, call number, title or room number. 16K, 32K or Commodore 64. *Multiple Choice* creates up to 150 questions and answer sets per disk file. Any number of questions can be selected and randomized. Test can be taken on screen or printed on paper. Provides high security for teacher-made tests. 16K, 32K or Commodore 64.
Price: Datalog and Date Due $39.95; Multiple Choice $29.95

Company:
Connecticut microComputer
36 Del Mar Drive
Brookfield, CT 06804
203-775-4595

Product:
BUSSter A64 Digital Input Module—IEEE-488 64 digital channel data acquisition input module. Accepts commands from any computer through its IEEE port, reads and stores data from up to 64 digital TTL level lines and then sends this information back to the computer. Economically increases a computer's interfacing capability while reducing its workload. Programmed through BASIC commands from the controlling computer. The built-in timer and buffer allow data sampling and collection to occur while the host computer is occupied with other tasks.
Price: $495.00



IEEE-488 64 DIGITAL CHANNEL DATA ACQUISITION INPUT MODULE

- 64 Digital Inputs
- Built in timer
- Buffered
- 2048 Byte Buffer Optional

Company:
Electronic Specialists, Inc.
171 South Main Street
Natick, MA 01760
617-655-1532

Product:
Interference Control and Electronic Products Catalog—A new 40-page catalog from Electronic Specialists that presents their line of computer interference control products. Protective devices include equipment isolators, AC power line filter/suppressors, line voltage regulators and AC power interrupters. Request catalog 831.

Company:
J.L. Hammett Company,
Microcomputer Division
P.O. Box 545
Braintree, MA 02184
800-225-5467 (In Mass. 800-972-5056)

Product:
1983 Hammett Microcomputer Catalog—A 48-page catalog offering over 300 administrative and instructional products, covering everything from computer literacy to library, school and classroom management to math, reading, science and art. Features educator evaluated software and other computer essentials.

Company:
Focus
Box 180
Stony Brook, NY 11790

Product:
Catalog of programs for educators—All programs are class-tested, process-oriented teaching aids for the PET and Commodore 64.
Price: Free

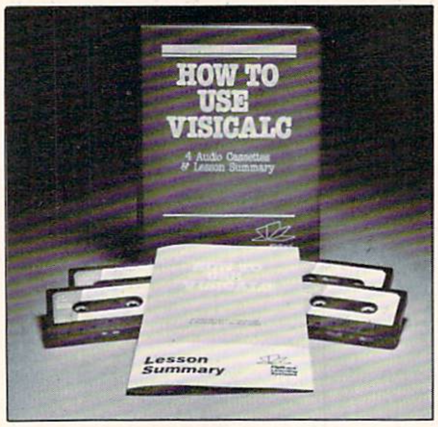
Company:
Educational Systems, Inc.
1000 Skokie Boulevard
Wilmette, IL 60091
312-256-4750

Product:
MICROREF™—Quick reference guides for today's most popular microcomputer software. Lists key procedures in easy-to-follow steps. Has a built-in easel and thumb-indexing for easy accessibility. Printed on nonglare plastic sheets for long life.
Price: $19.95 to $24.95

new products

Company:
Scholastic, Inc.
730 Broadway
New York, NY 10003
212-505-3000

Product:
Teaching and Computers—A new monthly magazine dedicated to helping elementary school teachers understand and use the microcomputer in the classroom. To be published eight times a year beginning in September, 1983.
Price: Introductory subscription price is $15.95 a year.



Company:
Fliptrack® Learning Systems
526 N. Main Street
Glen Ellyn, IL 60137
312-790-1117

Product:
How to Use VisiCalc®—Audio cassette tutorial that "talks" the learner through the development of a complex VisiCalc model. Starts with the basics of how to set up a spreadsheet then moves on to more advanced concepts.
Price: Contact company

VisiCalc is a registered trademark of VisiCorp

SimplexSoft Ltd.

FINANCIAL RECORD SYSTEM
For VIC 20™—Commodore 64™

- Record, total, and/or print all income sources
- Record, total, and/or print all expenses
- Record, total, and/or print all the items on the most complex US tax forms (car, travel, entertainment, supplies, contributions, medical, local & state taxes, etc.)
- Review & Edit all entries by day, week, month, year, and/or by category (car, medical, taxes, etc.)
- Store & Retrieve all Data on tape or disk

TAX TIME WILL TAKE AN HOUR OR SO NOT DAYS OR WEEKS

IDEAL FOR:
- Individual & Multi-Income family financial & tax records
- Self employed & company Reps. on expense accounts
- Apartment owners and managers
- Small Contractors (Bldg., Plumbing, Heating, Elect., Etc.)
- Truck owners/operators
- Farmers
- Any other small businesses

NO KNOWLEDGE OF COMPUTER LANGUAGE NEEDED—ALL IN PLAIN ENGLISH

VIC 20 version requires 16K memory expander
(Printer not required)
SimplexSoft 2 tape system only............................$29.95
Disk.. 34.95
Specify cassette or disk and computer model
Add $2.00 for mailing.

Send check or money order to:

SimplexSoft, Ltd.
617 N. Property Lane
Marion, Iowa 52302

DEALERS
DON'T MISS OUT—ANYONE CAN DEMO THIS SYSTEM AND LOOK PROFESSIONAL

VIC 20 and Commodore 64 are trademarks of Commodore Electronics Ltd.

advertisers index

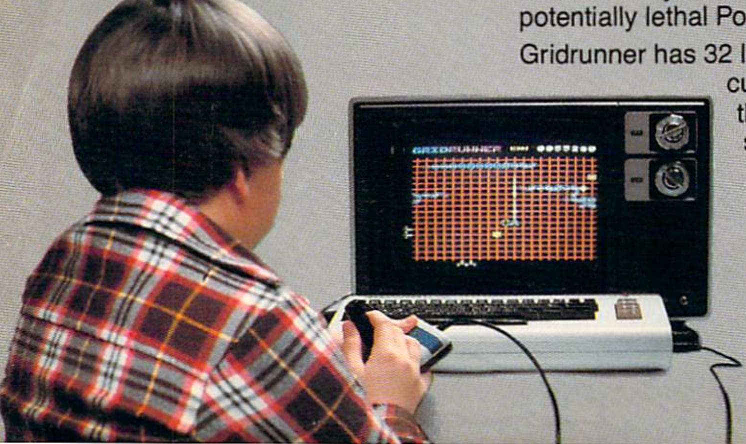
ABC Data Products	10
Academy Software	10
Advanced Arcadeware	105
American Peripherals	14
Arbutus Totalsoft	21
Commodore	32, 79
CompuSense	31
Computer Case Company	65
Computer Marketing Services	66
Connecticut microComputer	25
Cow Bay Computing	81
ETC.	69
Fox Soft	75
French Silk	29, 75
HES	IBC
Info Designs	IFC
Innovative Organizers	92
Input Systems	7
K-12 Micromedia	69
Leading Edge	OBC
LemData	12
Main Menu Programs	30
Micro 80	75
Microsignal	95
Micro Systems Development	8
Microware Distributors	6
Microworld Electronix	110
M.I.K., Inc. USA	69
New Leaf	73
O.E.M.	29
Pearr Desk Systems	100
Precision Technology	100
Professional Software	2
Pyramid	95
Rainbow Computer Corp.	21
St. Martin's Press	16
Simplex Soft Ltd.	120
Skyles Electric Works	22
Softwerx	13
Southern Solutions	1
Star Micronics	3
Toronto PET Users Group	110
TOTL Software	102
Virginia Micro Systems	105

IS GRIDRUNNER™ UNBEATABLE?

No one, not even the author, has ever achieved the last Gridrunner. It is an extremely fast-paced arcade-quality game designed to test your coolness under fire and challenge your reflexes.

As the pilot of the Gridrunner, a combat ship, you must annihilate the various enemies traveling along the "Grid." High scores are possible only through the mastery of the patterns of the X/Y Zappers and the Gridsearch Droids which, when destroyed, mutate into potentially lethal Pods.

Gridrunner has 32 levels of difficulty (20 levels in the VIC 20 version). To this date, the 13th level has been the highest achieved.



Gridrunner is available for VIC 20™ and Commodore 64™.

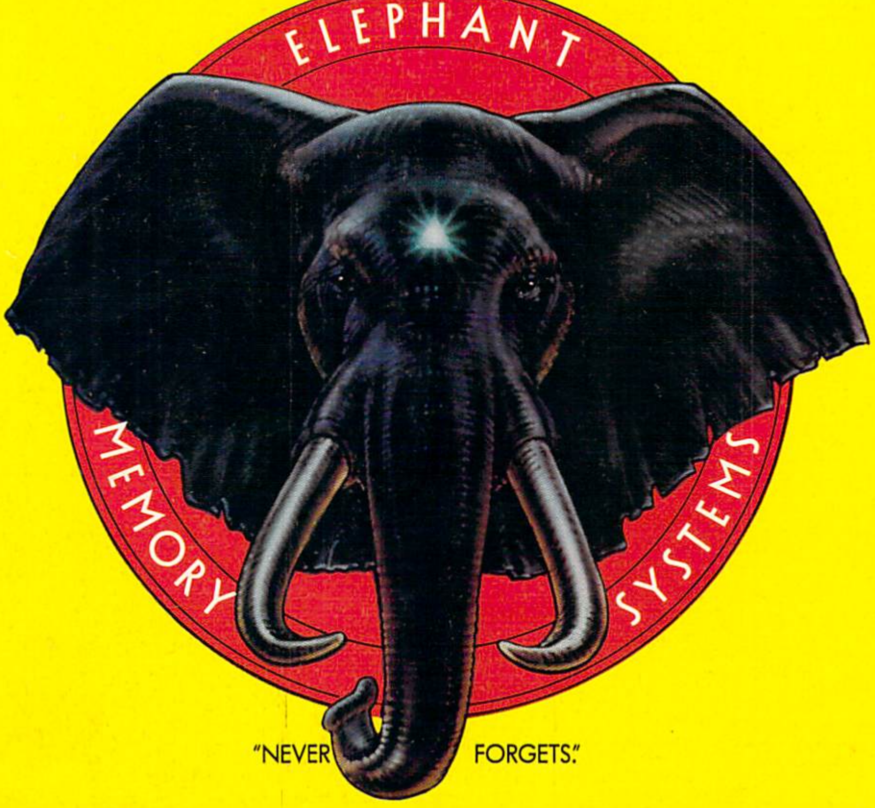
Can you beat Gridrunner? See your local computer or games dealer and find out.

Human Engineered Software
71 Park Lane
Brisbane, CA 94005

HES
a division of USI

VIC 20 and Commodore 64 are trademarks of Commodore Electronics Ltd.

REMEMBER:



MORE THAN JUST ANOTHER PRETTY FACE.

Says who? Says ANSI.

Specifically, subcommittee X3B8 of the American National Standards Institute (ANSI) says so. The fact is all Elephant™ floppies meet or exceed the specs required to meet or exceed all their standards.

But just who is "subcommittee X3B8" to issue such pronouncements?

They're a group of people representing a large, well-balanced cross section of disciplines—from academia, government agencies, and the computer industry. People from places like IBM, Hewlett-Packard, 3M, Lawrence Livermore Labs, The U.S. Department of Defense, Honeywell and The Association of Computer Programmers and Analysts. In short, it's a bunch of high-caliber nitpickers whose mission, it seems, in order to make better disks for consumers, is also to make life miserable for everyone in the disk-making business.

How? By gathering together periodically (often, one suspects, under the full moon) to concoct more and more rules to increase the quality of flexible disks. Their most recent rule book runs over 20 single-spaced pages—listing, and insisting upon—hundreds upon hundreds of standards a disk must meet in order to be blessed by ANSI. (And thereby be taken seriously by people who take disks seriously.)

In fact, if you'd like a copy of this formidable document, for free, just let us know and we'll send you one. Because once you know what it takes to make an Elephant for ANSI . . .

We think you'll want us to make some Elephants for you.

ELEPHANT.™ HEAVY DUTY DISKS.

For a free poster-size portrait of our powerful pachyderm, please write us.
Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021
Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.